LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

ÑO VI. 2ª ÉPOCA NÚMERO 60 UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L. 975 Ptas. •1.280 Esc-Cont • 5,86 € (IVA incluido)

NUEVA TECNOLOGÍA ASP

GRATIS
StarOffice 5.1 para Solaris
(Intel x86 y Sparc)
100% Funcional y sin límite de tiempo

Sun

NUEVAS SECCIONES
WINDOWS NT
LINUX

NS A FONDO (I)
ntroducción al protocolo

NCRIPTACIÓN (I)
undamentos de la criptografía

APTURA DE SECUENCIAS CON DELPHI (Y IV)
esarrollo de aplicaciones

INDOWS NT (I)
onocimientos a fondo e IIS 2.0

TRANSFERENCIA DE ARCHIVOS PUNTO A PUNTO (III)
Conclusión de la creación de un chat

VB: SUITE DE INTERNET (II)
Envío de correo electrónico

BBDD: DESARROLLO CLIENTE/SERVIDOR (IV)
Servidor de datos SQL Server

PROGRAMACIÓN DEL REGISTRO DE WINDOWS (Y III)
Creación de entradas en el registro

ONTENIDO CD

- Free Pascal Compiler 0.99.12b
- Visual Prolog P.Ed. 5.1
- JCloak 3.0
- Mozilla Source Milestone 9
- JFCSuite 2.0
- Registry Studio 1.01
- Jammer 1.95
- NetIntellect 4.0.1
- WinGate 3.0.4
- Understand for C++ 1.2.7

8 413042 303299 00060

SUSCRÍBETE AHORA Y OBTENDRÁS..

1. • El **pack completo COREL DRAW 8.0**. COREL DRAW, COREL DRAW PHOTO-PAINT 8, COREL OCR-TRACE 8, COREL DREAM 3D 8, COREL TUTOR, COREL CAPTURE 8, COREL TEXTURE 8, COREL SCAN 8, COREL SCRIPT EDITOR, BITSTREAM FONT NAVIGATOR, más de 40.000 imágenes y símbolos, 1.000 fotografías, 1.000 fuentes True Type y Tipo 1, 750 objetos y 450 plantillas de COREL DRAW.
2. • **50 CD-ROM** con más de 200 programas y utilidades de diseño gráfico, 500 fotografías profesionales y 49 Webs de diseño para navegación, necesarias para el seguimiento del máster junto con su código de ejercicios.
3. • **50 fascículos** coleccionables divididos en dos partes diferenciadas: Teoría y Práctica, que componen más de 1.000 páginas del más completo manual de diseño gráfico con COREL DRAW 8.0, abarcando el uso del color y tipografías, filtros, técnicas de compresión y diseño Web.
4. • **TUTOR ON LINE**, que resolverá todas las dudas que pueden ir surgiendo durante el máster.
5. • **DIPLOMA** acreditativo, de que se ha recibido y aprovechado el Máster COREL DRAW de Diseño Gráfico.
6. • **6 CARPETAS** para almacenamiento de fascículos y CD-ROM.

Con él aprenderás a:

- Escanear fotografías y realizar retoques fotográficos.
- Combinar todos los elementos gráficos en una publicación.
- Crear gráficos optimizados para Internet, mapas de imágenes, etc.
- Exportar imágenes para su uso en páginas Web.
- Utilizar todo tipo de software gráfico especializado.
- Usar correctamente las tipografías en folletos, posters, etc.
- Diseñar animaciones y realizar el rodaje final.
- Comprender los diferentes sistemas de postproducción e impresión.
- Crear y componer escenas en 3D.
- Emplear todos los dispositivos de hardware relacionados con el diseño gráfico.

... y mucho más.

BOLETÍN DE SUSCRIPCIÓN

OFERTA VÁLIDA SÓLO PARA ESPAÑA

SI, deseo suscribirme al **MASTER COREL DRAW**

CONDICIONES DE PAGO: (MARQUE LA OPCIÓN DESEADA)

❑ Pago al contado: A partir de la tercera entrega.
29.990 Ptas.
AHORRO con respecto a precio en quiosko: más de 14.000 ptas.
Se entregará una versión completa de Corel DRAW 8.0 en el primer envío

❑ Pago a plazos:
Primer envío: 16 primeros fascículos por **9.750** ptas.
Resto: 8 mensualidades de **3.250** ptas.
Corel DRAW 8.0 se incluirá en los envíos 7 y 12

Precio en el Quiosko: 795 Ptas. cada entrega de 1 fascículo y un CD-ROM
Total 50 fascículos, 50 CD-ROM y 6 tapas: 44.125 ptas.

UTILICE MAYÚSCULAS PARA RELLENAR ESTA TARJETA

Nombre y apellidos .....F. nacimiento.....

Domicilio ..... C.P. .....

Ciudad ..... Provincia .....Telf. .....

e-mail ..... Profesión .....

FORMAS DE PAGO:

❑ Con **cargo a mi tarjeta VISA** Nº ____ ____ ____ ____ Caduca .....

❑ **Domiciliación bancaria**

Sr. Dtor. del banco .....

Población .....

CÓDIGO CUENTA CORRIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

Ruego a UD. que se sirva cargar en mi ❑ cuenta corriente ❑ libreta de ahorro el/los recibos que le serán presentados por REVISTAS PROFESIONALES, S.L. como pago de mi suscripción al **MASTER COREL DRAW DE DISEÑO GRÁFICO**

❑ **Cheque** a nombre de REVISTAS PROFESIONALES, SL.

❑ **Contra reembolso** del importe más gastos de envío.

❑ **Giro Postal** (adjunto fotocopia del resguardo)*. Firma

*Esta forma de pago es exclusivamente para pagos al contado

SÓLO PROGRAMADORES

Número 60
SÓLO PROGRAMADORES
es una publicación de
REVISTAS PROFESIONALES S.L.

Editor
Agustín G. Buelta
Director
Javier Amado Buiza
Coordinador Técnico
Eduardo De Riquer Frutos
Coordinadoras de Redacción
Gema Romero Moreno-Manzanaro
Cristina Peña del Pozo
Colaboradores
Constantino Sánchez, Juan Luis Ceada, Jorge Delgado, Javier Sanz, Adolfo Aladro, Javier Toledo, Esteban Amado, Jordi Agost, Vicente A. Sánchez Werner
Maquetación y Tratamiento de Imagen
J. Santos, Alfonso Sabán
Consultas técnicas
atecnica@virtualsw.es

Publicidad
Paloma Seidel
Tel.: (91) 304 78 46
Mariano Sánchez (Barcelona)
Tel.: (93) 322 12 38
Pepín Gallardo (Barcelona)
Tel.: 617 09 36 68

Suscripciones
Rosa Tabares
Tel. (91) 304 87 64 Fax: (91) 327 13 03

Preimpresión
Ochoa Plus
Impresión
I.G.Pantone
Distribución
Motorpress Ibérica

La revista Sólo Programadores no tiene por qué estar de acuerdo con las opiniones escritas por sus colaboradores en los artículos firmados.
El editor prohibe expresamente la reproducción total o parcial de los contenidos de la revista sin su autorización escrita.

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 30-11-99
Precio en Canarias: 892 ptas. sin I.V.A.

EDITORIAL

El que no se arriesga...

Este mes hemos hecho un lavado de cara a algunas secciones, en especial destacar el sumario en el que hemos pretendido dejar aún más claro los temas que se pueden encontrar en el interior de la revista, para que todo os resulte más cómodo y directo.

Pero nuestra revista no es lo único que ha cambiado, como muchos de vosotros sabréis *Sun Microsystem* ha adquirido *StarOffice,* con lo que esta potente *Suite* ha experimentado un aumento en el número de descargas a través de *Internet*. Como siempre ha sido nuestra intención abarcar a todo tipo de público, este mes ofrecemos como segundo *CD-ROM* la versión para entorno *Solaris* bajo plataformas *Intel x86* y *SPARC*. Hemos de destacar que esta **versión es completa, 100% funcional y sin limitación de tiempo.** En las páginas 20 y 21 os mostramos el proceso de instalación para aquellos que quieran contar con esta fantástica aplicación.

Este mes hemos decidido incorporar dos secciones nuevas en nuestro contenido. La primera es la sección de *Windows NT*. Con ella queremos abarcar un espacio del complejo e interesante mundo de los sistemas servidores, del cual *NT* (especialmente en nuestro país) forma parte muy importante de las pequeñas, medianas y grandes empresas. En este primera parte se tratarán los principales conceptos y las diferencias entre los clásicos *Windows 95/98* y *NT*. También hablamos de *IIS 2.0* y del *Option Pack 4*. Con estos conceptos claros posteriormente comenzaremos a trabajar sobre características más concretas tanto del sistema como de los paquetes que trabajan bajo este entorno.

La segunda novedad que nos abarca este número es la incorporación del tan nombrado *Linux*. Retoma su posición en nuestra revista con el fin de complacer a todos los lectores que lo echaban de menos. Comenzaremos esta sección con un reportaje sobre el *DNS*, que aunque abarca tanto a entornos *Windows* como *Linux* es, y ha sido, más utilizado en el segundo. Estos conceptos son la base de las direcciones de red y más aún de *Internet,* por eso empezamos explicando los conceptos y las bases de este protocolo.

Además, y como podréis comprobar este número está plagado de temas nuevos, os destacamos la serie que iniciamos sobre encriptación en *Java*, un mundo apasionante en un lenguaje cada vez más utilizado. Y como no, la tecnología *ASP* que tan presente está últimamente en *Internet* y a la que hemos querido dedicar la portada de este mes.

Tampoco nos olvidamos de las series que continúan como, la creación de nuestra propia suite de *Internet*, la explicación sobre como realizar una correcta transferencia de archivos o la programación del Registro de *Windows*.

Ya sabéis lo que se dice últimamente: "El que no se arriesga no cambia y el que no cambia, nada es", por eso nosotros cambiamos mes a mes y número a número.

Javier Amado
DIRECTOR

SUMARIO 60

SÓLO PROGRAMADORES

6 Noticias

NOVEDADES
Como siempre en esta sección encontraréis las noticias más importantes de todo lo relacionado con la programación. Para que os mantengáis bien informados.

9 Contenido del CD Rom

Además de los programas y las actualizaciones más importantes de la Programación os incluimos también todos los listados que apoyan a los reportajes que encontraréis en la revista. Este mes lo más destacado de nuestro CD es: Free Pascal Compiler 0.99.12b, Visual Prolog Personal Edition 5.1, Mozilla Source Milestone 9, NetStudio 2000...

22 Java

ENCRIPTACIÓN (I)
Hoy más que nunca es necesario disponer de medios de transmisión seguros, por eso la criptografía se ha vuelto cada vez más importante. A lo largo de la serie de artículos que ahora iniciamos se mostrarán sus fundamentos, los métodos desarrollados y su correspondiente aplicación.

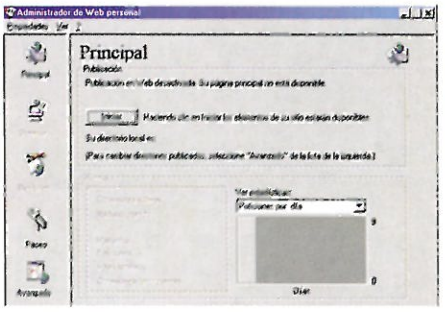
30 Vídeo

CAPTURA DE SECUENCIAS CON DELPHI (Y IV)
Para finalizar esta serie de artículos dedicados a la captura de secuencias de vídeo con Delphi desarrollaremos dos pequeñas aplicaciones, una de videoconferencia y otra que nos permitirá detectar el movimiento.

38 Linux

DNS A FONDO (I). INTRODUCCIÓN AL DNS
El servicio de DNS es uno de los más importantes protocolos que permiten el funcionamiento de Internet pero un gran desconocido para la mayoría del público. En esta serie de artículos que ahora iniciamos lo analizaremos con profundidad desde dos puntos de vista, el del usuario y el del servidor.

12 Internet

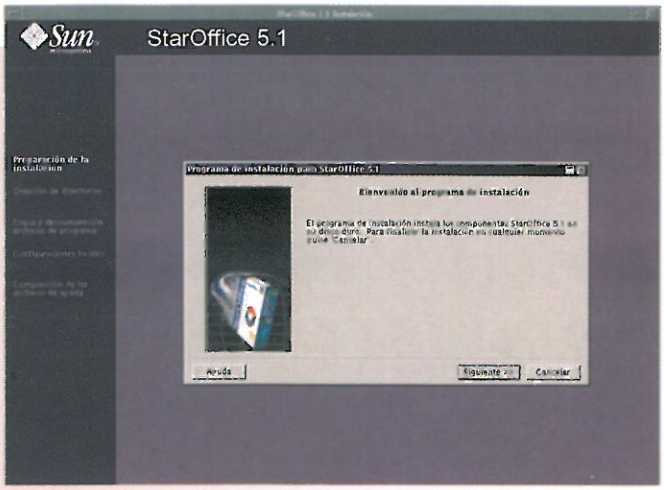


LA TECNOLOGÍA ASP (I)
Las páginas *ASP* constituyen una manera excelente de afrontar las aplicaciones *Web*. Su sencillez y economía han hecho de ellas una de las tecnologías dominantes, en la que nosotros nos adentraremos en éste y en los próximos artículos.

20 CD-ROM ESPECIAL

STAROFFICE 5.1 PARA SOLARIS 7

No hemos querido dejar escapar la oportunidad de incluiros un CD extra, dedicado exclusivamente a la nueva versión de StarOffice. Y para que todo os resulte más fácil en estas páginas encontraréis una explicación rápida y sencilla de sus características e instalación.



46 WINDOWS NT

WINDOWS NT 4.0 e IIS

Las siglas *NT* suenan cada vez más en todos los ámbitos, ¿Qué es? ¿Qué diferencias tiene respecto a *Windows 95/98*? ¿Cuántas versiones existen? ¿Cuáles son sus diferencias? Eso es lo que intentaremos explicar en las próximas páginas. Y como este sistema operativo no podía faltar en nuestras páginas, a partir de ahora encontraréis esta sección de forma más o menos habitual, en la revista.

52 REDES

TRANSFERENCIA DE ARCHIVOS PUNTO A PUNTO (III)

En esta tercera parte terminaremos el componente que se encarga de la transmisión de datos por el puerto serie, habrá que implementar un programa para probar su correcto funcionamiento. La aplicación que hemos elegido es un *chat* a través del puerto serie, es decir, una aplicación que permite transmitir mensajes de texto entre dos ordenadores.

58 VISUAL BASIC

SUITE DE INTERNET (II): EL CORREO ELECTRÓNICO (I)

En el inicio de esta serie vimos cómo incorporar un navegador, con todas las posibilidades que éste ofrece, a nuestro proyecto. Ahora vamos a ver cómo incorporar el que quizás constituya el mayor uso de *Internet* hoy en día: el correo electrónico.

66 BASES DE DATOS

DESARROLLO CLIENTE/SERVIDOR (IV)

En el número anterior enfocamos el sistema cliente/servidor desde el punto de vista de un servidor de base datos *Oracle*. Ahora exponemos los mismos puntos de vista, pero para un servidor de datos *SQL Server*, adelantando el proceso de creación de la BBDD que se verá en el siguiente número.

72 WINDOWS 95/98

PROGRAMACIÓN DEL REGISTRO (Y III)

En este último artículo vamos a tratar todas aquellas funciones que permiten escribir en el Registro de *Windows*. Así lograremos que nuestras aplicaciones tengan un funcionamiento más profesional. Utilizaremos el Registro como una zona de almacenamiento permanente de datos de configuración.

78 LIBROS

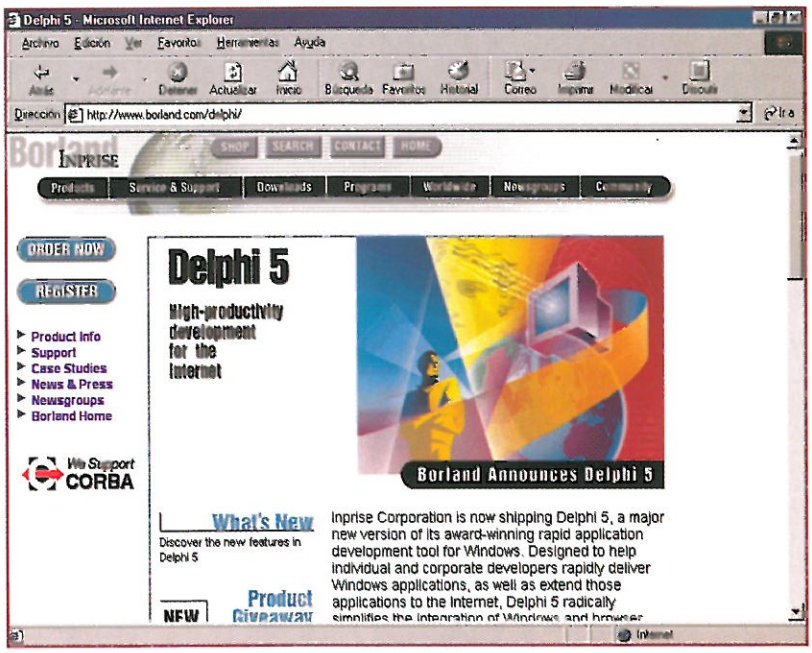
Las últimas novedades editoriales sobre programación y los libros que nos han parecido más interesantes aparecen en estas páginas con una sencilla reseña que os ayude a conocerlos y a decidir.

80 DUDAS TÉCNICAS

Como siempre estamos dispuestos a contestar a todas vuestras dudas sobre programación. No os preocupéis si os parecen demasiado sencillas o demasiado complejas, siempre podréis acudir a nuestra dirección de e-mail: **solop@virtualsw.es.**

DELPHI 5 YA ESTÁ EN ESPAÑA

El pasado día 22 de septiembre *Borland* presentó en Madrid la nueva y esperada versión de *Delphi*. La potente aplicación de herramientas de desarrollo para *Windows* mejora sus capacidades aumentando la rapidez del desarrollo tanto para desarrolladores individuales como corporativos, extendiendo sus aplicaciones a *Internet*. Además, el nuevo *Delphi* 5 soportará a partir de ahora tanto *HTML* 4 como *XML*.



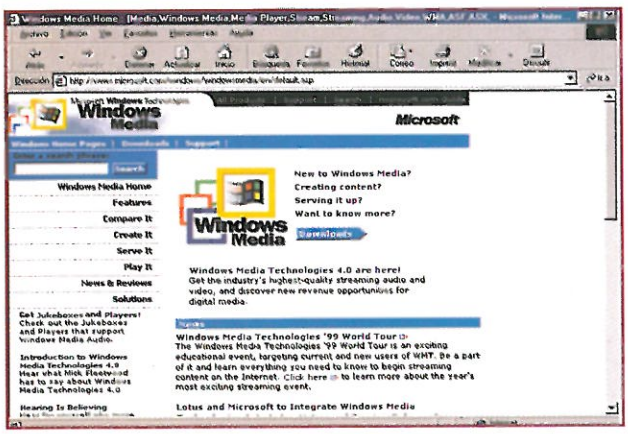
Todo ello simplificará, según sus creadores, la integración de *Windows* con los buscadores, los clientes, los servidores *Web*... y en general todo lo relacionado con el mundo de *Internet*. Incluyendo un rápido acceso a las bases de datos tanto relacionales como no relacionales, al correo electrónico e incluso al sistema de ficheros.

Además de soportar *HTML* 4 y *XML*, el nuevo Delphi 5, al utilizar *ActiveX Data Object* (*ADO*) de *Microsoft*, permite un rápido acceso a todo tipo de información. Una aplicación diseñada para el comercio electrónico.

Entre las mejoras que introduce Delphi 5, desde *Borland* destacan el equipo fuente (*TeamSource*) como un forma de incrementar la productividad de desarrollo, sobre todo operando con el código fuente. Además de incluir *Visual Component Library (VCL)* que simplifica los procesos más complejos del desarrollo de aplicaciones, tanto de *Internet* como de Windows. Sin embargo, y pese a su presentación oficial en España, todavía no hay una fecha exacta para que *Delphi* 5 esté disponible para el público en general. Podéis encontrar más información en **http://www.borland.com/delphi/**

MICROSOFT PRESENTA WINDOWS MEDIA TECHNOLOGIES 4

La compañía *Microsoft* dió a conocer la plataforma multimedia más innovadora disponible hoy en día: *Windows Media Technologies* 4 que incluye una avanzada tecnología de audio que aporta un nuevo estándar para audio de calidad *CD* en *Internet*.



Al contrario que otros productos de esta categoría, *Windows Media Technologies* 4, logra una calidad *CD* de audio con la mitad del tamaño del formato. Además, ofrece un buen número de ventajas entre las que se encuentra la tecnología de *streaming* sin coste, por la que se rompe definitivamente esta barrera; el vídeo, con calidad cercana a la televisión, mejora la calidad de la imagen en conexiones de baja velocidad, o la solución de gestión de derechos digitales pudiendo el editor gestionar y controlar la distribución y uso del contenido de audio y vídeo digitales.

Según sus declaraciones las compañías líderes en este sector han quedado satisfechas con los resultados obtenidos con este nuevo producto de *Microsoft*.

El *Windows Media Technologies* es la principal plataforma de medios digitales para suministrar a los consumidores y compañías una calidad de audio y vídeo sin precedentes. Esta herramienta se encuentra disponible para ser bajado de Internet en **http://www.microsoft.com/windows/windowsmedia/**

ALIANZA SUN-NETSCAPE CON SU NUEVA MARCA CONJUNTA: IPLANET

La alianza *SUN-Netscape* ya es una realidad, así se hacía patente durante los primeros días del mes de septiembre.

Con esta nueva recién nacida unión se beneficia el liderazgo tecnológico y la economía en *Red* entre otros aspectos, convirtiéndose en los líderes del sector de las tecnologías de la información.

Los productos *software* de infraestructura que ofrecen ambas empresas aportan la escalabilidad, la fiabilidad y el rendimiento necesario para soportar el enorme volumen de tráfico existente en *Internet*.

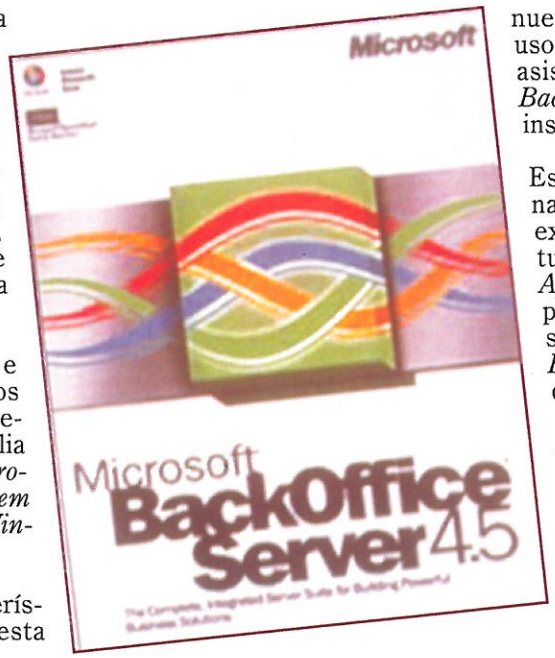
A la vez que se daba a conocer la alianza, también presentaban su nueva marca de productos y servicios *iPlanet*, cuya estructura de *software* es una de las más completas e integradas del mercado, ofreciendo Servidores de Aplicaciones, Servidores *Web*, Servidores de Directorios y Seguridad, Servidores de Mensajería y Colaboración y Acceso Remoto.

Todas estas aplicaciones permiten a las empresas reaccionar ante las oportunidades de negocio que ofrece la Economía en *Red*, sin incurrir en los costes de desarrollo y el mantenimiento del *software* a medida.

Para buscar más información sobre el tema sólo basta con visitar el sitio *Web*: **www.iplanet.com**

NUEVA VERSIÓN DE BACKOFFICE SERVER

Recientemente la compañía *Microsoft* anunciaba la presentación de *BackOffice Server* 4.5, una *suite* integrada de soluciones servidor, optimizada para el sistema operativo *Windows NT Server*, que se encarga de simplificar la implantación, gestión y desarrollo de potentes soluciones para grandes empresas.



Esta herramienta aporta e integra la instalación de los últimos componentes y paquetes de servicios de la familia *BackOffice*, incluyendo *Microsoft SQL Server* 7, *System Management Server* 2.0 y *Windows NT Service Pack* 4.

Entre algunas de las características más destacables de esta nueva versión, podemos resaltar el uso de *Setup script* reutilizables, los asistentes para la implantación del *BackOffice Server* o el escenario de instalación para sucursales.

Esta moderna aplicación proporciona la manera más sencilla para extender y gestionar una arquitectura *Windows Distributed Internet Applications*, proporcionando componentes clave de aplicaciones para servidor, incluyendo un sitio *Web*, *E-mail*, bases de datos, servidores de búsqueda de información, etc.

BackOffice Server ayuda a comenzar la implantación de *Internet* instalando automáticamente un *site intranet* de comienzo completo con siete aplicaciones para empezar, que incluyen desde noticias hasta aplicaciones de ayuda al escritorio.

AZLAN, CISCO Y UNISORCE LANZAN UNA OFERTA CONJUNTA EN REDES PRIVADAS VIRTUALES IP

Las compañías *Unisorce* y *Cisco* anunciaron el mes pasado la creación de una oferta conjunta de equipos de comunicaciones y servicios de red privada virtual *IP* que será comercializada por *Azlan*.

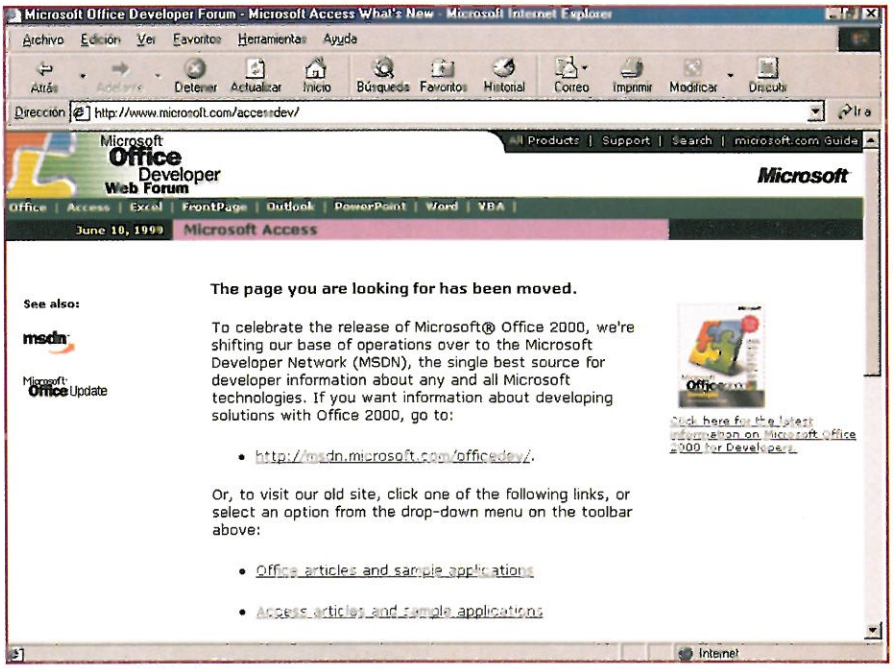
Este acuerdo está especialmente orientado a aquellos clientes entre cuyas necesidades figura el acceso a *Internet* desde una red de área local *(LAN)* o la interconexión de diferentes redes de este mismo carácter.

Esta oferta cuenta con el soporte de *Azlan*, la empresa encargada de la comercialización, así como de prestar apoyo al integrador de sistemas en toda la fase de configuración, y soporte de la instalación. Con la experiencia de esta compañía esperan reducir los problemas al mínimo.

Las tres compañías coinciden en que esta oferta simplifica la adquisición de los productos y servicios. De este modo el usuario sólo tiene que preocuparse por la utilización de la infraestructura para conseguir sus objetivos de negocio, obteniendo un importante ahorro en los costes.

NUEVAS HERRAMIENTAS FACILITAN LA MIGRACIÓN DE ACCESS A SQL SERVER 7 DE FORMA GRATUITA

Microsoft ha anunciado una promoción especial por tiempo limitado, así como una serie de herramientas de migración que hacen que los usuarios encuentren más fácil que nunca la migración de sus aplicaciones *Microsoft Access* a *SQL Server 7*.



Para facilitar este proceso se han integrado las herramientas *Microsoft Access Upsizing Tools* directamente en *Access 2000*; con lo que se logra que éstas conviertan de forma automática los objetos del *Microsoft Access* a sus equivalentes en *Microsoft SQL Server*. De este modo, los desarrolladores podrán conservar la estructura de sus bases de datos, incluyendo los datos, índices, ajuste por defecto y contadores. Además mantiene las relaciones de las tablas y la integridad de las referencias después de realizada la actualización.

Los usuarios que deseen migrar directamente a *SQL Server* pueden utilizar las *Access 97 Upsizing Tools* que se encuentran disponibles para ser descargadas de la *Web* de forma gratuita (sólo hay que pagar los gastos de conexión) en la dirección: **http://www.microsoft.com/accessdev**

HERRAMIENTAS DE PROGRAMACIÓN

ENTORNOS DE DESARROLLO

BOXER 99 V8.0.0

Completo editor de texto. Incluye opciones de configuración, comparación de ficheros, búsquedas, corrección ortográfica, etc... e incluso permite definir programas externos como herramientas.

FREE PASCAL COMPILER 0.99.12B

Compilador de *Pascal* de 32 *bits*. Su sintaxis es compatible con *Turbo Pascal* 7.0 y algunas extensiones utilizadas por *Delphi*. Está disponible para *DOS*, *Win32* y *Linux*. Incluye el código fuente y documentación del compilador en formato *TXT*, *PDF*, *PS* y *HTML*.

HYPERTEXT STUDIO 3.0

Herramienta de autor para publicar páginas *HTML* y ficheros de ayuda de *Windows*. Su entorno de desarrollo visual facilita la creación de los documentos. Incluye herramientas de gestión de proyectos, una base de datos para controlar y almacenar la información, un verificador de la ortografía.... Puede importar ficheros *HHP*, *HPG*, *HTM*, y *RTF* y publicar las páginas *Web* en *Internet*.

ULTRAEDIT PROFESSIONAL TEXT/HEX EDITOR 6.20B

Potente editor de texto y hexadecimal para profesionales. Trabaja con múltiples ventanas, maneja ficheros de cualquier tamaño y puede visualizar los ficheros en el navegador que utilicemos habitualmente. Incluye plantillas, un verificador de la ortografía en varios idiomas, opciones de formato, soporte para macros, funciones de búsqueda y sustitución, etc. Convierte ficheros entre formatos *DOS* y *UNIX*.

VISUAL PROLOG PERSONAL EDITION 5.1

Entorno de programación para el desarrollo de aplicaciones en lenguaje *Prolog*. Ofrece todo lo necesario para construir grandes aplicaciones incluyendo un compilador y una librería con enlaces a *API´s* especializadas. Permite la edición gráfica de diálogos, menús y barras de herramientas. Puede trabajar con bases de datos *ODBC* y *OCI*. También ofrece soporte para *Internet* para *sockets*, *FTP*, *HTTP*, *scripts CGI*, etc.

LENGUAJES

VISUAL BASIC

ADOCUMENT! VB 1.2

Agiliza el proceso de creación de documentación de ayuda *HTML* para proyectos en *VB*. Su avanzado analizador estudia el código fuente para crear un conjunto estructurado de páginas de ayuda compiladas en un fichero *HTML*. Además de crear el documento, puede añadir ayuda sensible al contexto.

MP3 MAKE - ENCODER OCX 1.1

Control *ActiveX* que convierte ficheros *WAV* a formato *MPEG layer* 3. Puede utilizarse dentro de todos los programas de desarrollo que soporten controles *ActiveX/OCX*. Ofrece una compresión de 12 a 1 sin que la pérdida de calidad sea significativa. Incluye código fuente de ejemplo en *Visual Basic* y documentación.

MP3 PLAY - DECODER OCX 1.5

ActiveX que puede convertir ficheros *MP3* a *WAV*. Trabaja en todos los programas de desarrollo que soporten controles *ActiveX/ OCX* como *VB*. Incluye código fuente de ejemplo en *VB* y documentación.

TIMETOWIN 7.25

Colección de 970 funciones y subrutinas para programadores de *Visual Basic*. Recoge rutinas de: geometría, *bitmaps*, claves del Registro, compresión, *CRC*, encriptación, ficheros, cadenas, correo electrónico, *RAS*, noticias, trabajo en *Red*, etc.

JAVA

CLOAK 3.0

Protege aplicaciones *JAVA*. Evita que personas no autorizadas puedan acceder y utilizar el código fuente de los programas desarrollados. Aunque no impide que sean descompiladon, el resultado es ininteligible. Protege *applets*, aplicaciones y librerías, crea ficheros ejecutables *JAR* y reduce el tamaño de los *applets*.

JFCSUITE 2.0

Colección de *JavaBeans* que añade nuevas funciones al estándar *Java Foundation Classes*. Incluye un calendario para aplicaciones de gestión de información personal (*PIM*) y mejoras para la edición de monedas, fechas y horas, entre otros.

HTML

MERACL IMAGEMAP GENERATOR 3.01

Genera mapas de imágenes para un sitio *Web*, es decir, imágenes que inician eventos. Suele llevar de una página a otra, pero también es posible que ejecute *JavaScripts*.

TOPSTYLE 1.01

Shareware – 2 MB - W95/W98/NT
Potente editor de hojas de estilo *CSS*. Cuenta con todas las herramientas para crear plantillas de estilo como un verificador de código de color, un visor interno para previsualizar el trabajo..., además, con él es posible utilizar *Internet Explorer* y *Mozilla Gecko*.

WEBINET 3.02
Herramienta de autor que añade movimiento a las páginas *Web*. Ofrece un modo de trabajo *WYSIWYG* y no requiere contar con conocimientos de *HTML* o *Java*. Inserta texto, imágenes, sonidos, *banners* y vídeo; crea presentaciones y sitios *Web* en tres idiomas, manteniendo la misma estructura en las páginas.

OTROS

MOZILLA SOURCE MILESTONE 9
Código fuente en *C/C++* de *Mozilla*. Nueva versión de desarrollo de este popular navegador de *Internet*. El código fuente está cubierto por la licencia pública de *Netscape*.

SOCKETTOOLS LIBRARY EDITION 3.10.3128
Conjunto de controles *VBX* y *ActiveX* para el desarrollo de aplicaciones para *Internet* e *intranets*. Incluye: *Domain Name Server*, *Web Browser Control*, un cliente *Telnet*, *POP3*, *FTP*, *Remote Execution* (*RLIB*), *SocketWrench* (*WinSock*), *Ping*, *Image Viewer*, *Terminal Emulation*, *SMPT*, *NNTP* y *RAS*.

UNDERSTAND FOR C++ 1.2.7
Analiza un proyecto en *C++* para facilitar la elaboración de su documentación. Muestra con rapidez las clases, las jerarquías, las llamadas, etc. y cómo y dónde está siendo utilizado el código fuente. Además ayuda a comprender mejor las aplicaciones creadas.

HERRAMIENTAS Y UTILIDADES

AUTOMATE PRO 4.3E
Aplicación de programación de tareas, avisos y lenguaje de macro. Permite automatizar cualquier tarea repetitiva como ha-cer copias de seguridad, transferir ficheros de *Internet*, realizar informes, etc. Tareas que pueden implicar a otras aplicaciones a la vez, *BBDD* de *Internet*, sistemas y ficheros.

BUTTONWIZ 6.3
Facilita la creación de botones para páginas *Web*. Incluye 200 estilos diferentes para los botones, cada uno con un atractivo color de fondo. Sólo hay que seleccionar un estilo, el color, el tipo de fuente e introducir un texto. *ButtonWiz* genera automáticamente una imagen que puede ser copiada al portapapeles o guardada en un fichero tipo *PNG*, *BMP*, *TIF*, *JPG*, *PCX* o *TGA*.

CHIEF'S INSTALLER PRO 4.77
Crea programas de instalación para aplicaciones *Windows* de 16 y 32 *bits*. Sus opciones permiten realizar instalaciones completas, parciales, personalizadas y programas de desinstalación. Trabaja con archivos *ZIP*, comprime y descomprime ficheros con *Microsoft COMPRESS.EXE*, y puede dividir ficheros para su inclusión en disquetes.

HYCD SUITE IN JAVA 2.13
Graba *CD's* con nuestras canciones favoritas. Permite copiar *CD's* completos, realizar mezclas de temas, borrar canciones, importar ficheros *WAV*, etc. Es muy fácil de usar y realiza automáticamente la configuración del sistema y los dispositivos de grabación.

NETSTUDIO 2000
Diseña elementos gráficos para sitios *Web*. Facilita la creación de textos, botones, barras de navegación, *banners*, etc. Incluye efectos y otras funciones de retoque. Puede incluso combinar varios objetos en una misma imagen, manteniendo la capacidad de editarlos por separado. Es totalmente compatible con *Microsoft FrontPage* 2000 y *Microsoft Office* 2000.

REGISTRY STUDIO 1.01
Añade nuevas funciones al editor del Registro de *Windows*. Se integra en el *RegEdit* para incorporar, entre otras interesantes características, una nueva barra de herramientas, botones de navegación, soporte para el procedimiento de arrastrar y soltar, un asistente y funciones de búsqueda en segundo plano.

RESOURCE-GRABBER 2.4A
Busca y extrae elementos multimedia para su utilización en otras aplicaciones. Localiza en los programas instalados en el ordenador *bitmaps*, imágenes para botones, iconos, sonidos y cursores. Soporta numerosos formatos como *HTML*, *CSS*, *RTF*, *BAT*, *TXT*, *COM*, *EXE*, *TLB*, *GIF*, *JPG*, animaciones, vídeo y ficheros de sonido.

REDES

EDEVICELOCK 3.2.3
Ofrece control sobre el acceso a discos duros y otros dispositivos de la *Red*. Permite a los administradores de redes asignar accesos a puertos *COM*, *LPT*, *CD-ROM's*, etc. Además realiza otras funciones que facilitan el seguimiento y control de todos los dispositivos.

JAMMER 1.95
Protege nuestra red y equipo de *NetBus* y *Back Orifice*. Detecta el acceso no autorizado al ordenador controlando todo el tráfico de nuestro equipo con otros ordenadores de una *Red*. Controla todos los servicios, puertos y protocolos.

NETINTELLECT 4.0.1
Herramienta de análisis de la actividad de un servidor. Realiza análisis históricos de tendencias, ofrece la posibilidad de combinar datos procedentes de diferentes servidores y cuenta con más de 50 opciones de filtro para realizar los informes. Incluye un módulo *FTP* interno y un gestor de ficheros que puede descomprimir archivos *ZIP* y *GZ*.

REMOTE ADMINISTRATOR 1.11
Aplicación de acceso remoto que nos permite trabajar en tiempo real con otro equipo. La pantalla del ordenador remoto es visualizada en una ventana en nuestro equipo de modo que simplemente podemos utilizar nuestro teclado o ratón para trabajar en el otro equipo.

RETINA 1.0
Ayuda a identificar y superar posibles problemas de seguridad de sistemas de *extranet*, *intranet* e *Internet*. Incluye módulos para verificar: *NetBIOS*, *HTTP*, *CGI*, *WinCGI*, *ISAPI*, *ASP*, *FTP*, *DNS*, *DOS*, *POP*, *SMTP*, *LDAP*, *TCP/IP*, *UDP*, Registro, servidores de bases de datos *SQL* y muchos otros.

WINGATE 3.0.4
Conecta a los usuarios de una *Red (LAN)* a *Internet* utilizando una única conexión. Permite acceder a grupos de noticias, *FTP*, correo electrónico, etc. sin necesidad de disponer de una dirección *IP* ni de un dominio.

Z-MONITOR 2.0
Analiza los datos enviados y recibidos a través de red y módems. Muestra todos los datos transmitidos en la red local o en *Internet*. Realiza un registro de las conexiones que se especifiquen y puede reconocer 21 protocolos diferentes.

DOCUMENTACIÓN/ TUTORIALES

APACHE SERVER FREQUENTLY ASKED QUESTIONS
Información sobre uno de los servidores más populares utilizados en *Internet*. Recordemos que *Apache* es el servidor *HTTP* utilizado en más de un 50% del total de los sitios *Web* de *Internet* ya que ofrece rapidez, estabilidad y potencia.

DVD FREQUENTLY ASKED QUESTIONS
Las preguntas mas frecuentes sobre *DVD*. con las cuestiones más interesantes planteadas en los grupos de noticias *REC.VIDEO.DVD*.

FAT32 AND HARD DISK PARTITIONING GUIDE
Tutorial que explica las *FAT* (*File Allocation Tables*). Ofrece una amplia información sobre *FAT32* y *FAT32x*, cómo aumentar el rendimiento en *FAT32*, conversión, etc. y, además, explica los comandos *FORMAT* y *FDISK*.

IRT.ORG FREQUENTLY ASKED QUESTIONS
Recopilación de las cuestiones más interesante sobre temas relacionados con la edición de sitios *Web*. Ofrece más de 1.400 respuestas a preguntas sobre *CGI* y *Perl*, hojas de estilo *(CSS)*, *Aural CSS*, *ColdFusion*, *HTML*, *HTML Dinámico*, *Java*, *JavaScript*, *VBScript*, realidad virtual *(VRML)* y *Extensible Markup Language* (*XML*).

MODEM FREQUENTLY ASKED QUESTIONS
Recoge interesantes consejos para obtener el máximo rendimiento de un módem de alta velocidad. Aquí se explican diversos temas para las conexiones a través de módem como optimización del sistema, normas, conexiones analógicas, cadenas de inicialización, etc.

MYSTERY BEHIND THE WINDOWS REGISTRY
Guía de información sobre el Registro de *Windows*. Explica qué es el Registro, cómo se estructura, los procedimientos para prevenir problemas y cómo recuperar este fichero en caso de fallo, trucos, etc.

IMPRESCINDIBLES

ANTIVIRUS
AntiViral Toolkit Pro 3.0.129
McAfee VirusScan 4.0.3
Panda Antivirus 6.07 Platinum

GRAFICOS
Icon Bank 4.0 Gold Edition
Paint Shop Pro with Animation Shop 6.0 beta 5
SureThing CD Labeler 2.0
ThumbsPlus 4.02
Xara3D 3.04

INTERNET
Añadir Pro 4.00.006
Copernic 99 v3.02
Cuentapasos 3.75
CuteFTP 3.0
Dial-Up Magic 1.8
Eudora Light 3.0.6
Go!Zilla 3.5
HomeSite 4.01
ICQ 99b beta 3.19 build 2569
PGP 6.0.2i
URL Organizer 2.3
WebTrends Enterprise Suite 3.6b

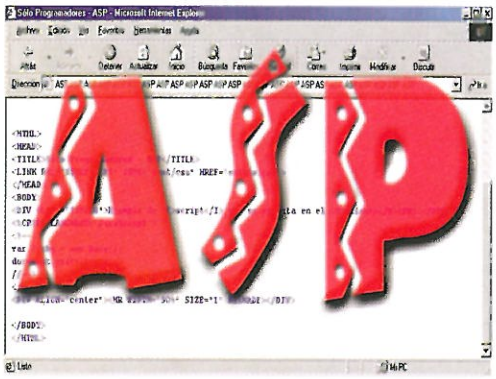
MULTIMEDIA
AudioCatalyst 2.01
COWON Jet-Audio 4.6
CDH Media Wizard 4.05
Sonique 1.05
WinAMP 2.50c

NAVEGADORES
Internet Explorer 5.0.2314
Netscape Communicator 4.61
Opera 3.60

PROGRAMACIÓN
Hackman 3.21
Help & Manual 2.1.1
InstallConstruct 3.2
Java Development Kit 1.2.2
Windows Registry Guide 1.2
WinEdit 32I-99e
WinHex 8.56

UTILIDADES
3DMark 99 Max b200
Adobe AcrobatReader 4.0
Advanced Registry Tracer (RegFix) 1.0a
Babylon Translator 2.1
CDRWin 3.7e
Emergency Recovery System 8.71
Nero 4.0.5.0
SiSoft Sandra 99 v5.10
Windows Commander 4.01
WinZip 7.0

ATENCIÓN:
En caso de problemas con el CD-ROM envíelo por correo ordinario, a la atención del SERVICIO TÉCNICO DE SÓLO P., incluyendo en el interior del sobre sus datos personales, a la siguiente dirección:
C/ San Sotero, Nº 5, 1º Planta,
28037 (Madrid)



La tecnología ASP (I)

Adolfo Aladro García (aaladro@arrakis.es)

Las páginas *ASP* constituyen una manera excelente de afrontar las aplicaciones *Web*. Su sencillez y economía han hecho de ellas una de las tecnologías dominantes. Presentan un modelo de programación integrado a la perfección con otras dos tecnologías actualmente punteras: *XML* y *DHTML*.

Las páginas *ASP* representan una tecnología que une sencillez, prestación y eficacia a la hora de desarrollar soluciones orientadas a la *Red*. Esta nueva serie tiene como propósito adentrarnos en la programación *ASP*. Pero queremos ir mucho más allá. El *HTML Dinámico* y el lenguaje *XML* han irrumpido con fuerza en estos últimos meses y las páginas *ASP* proporcionan una manera excelente de aunar todas estas tecnologías para dar lugar a sitios *Web* atractivos y funcionales.

PROGRAMACIÓN CGI Y ASP

Cada vez hay más nombres poblando el mundo de las aplicaciones *Web*. Antes de afrontar el aprendizaje de cualquier tecnología conviene detenerse por un momento para situarla con respecto a las demás. De esta manera se logra tener una visión global suficiente como para decidir en cada caso cuál es la más adecuada.

Las páginas ASP se fundamentan en la interfaz CGI

Las páginas *ASP* suponen un avance con respecto a la programación *CGI* tradicional. Aunque en los dos casos el sustento sigue siendo la interfaz *CGI* (*Common Gateway Interface*) son también muchas las diferencias existentes. Hasta no hace mucho tiempo la única forma que existía de recibir formularios o de generar dinámicamente páginas *Web* que presentaran información de una base de datos, era mediante programas hechos en *C*, *Perl* o cualquier otro lenguaje. Estos programas se desarrollaban a partir de la interfaz *CGI*, que se define como una serie de reglas que deben cumplir las aplicaciones cliente y servidor para hacer posible la presentación de resultados en tiempo real a través de la *Red*.

Al principio los programas *CGI* constituyeron toda una revolución pero pronto se revelaron como un mecanismo insuficiente. Entonces empezaron a surgir diversas tecnologías que facilitaban la labor de los programadores proporcionando una manera de acceder a la interfaz *CGI* al más alto nivel.

Las páginas ASP son ejecutadas por el servidor

Las páginas *ASP* constituyen en ese sentido todo un avance, pero es que además implican una nueva

forma de trabajar. El código que antes se ejecutaba en los servidores en programas independientes se encuentra ahora, con las páginas *ASP*, incrustado en mitad de las páginas. Éstas son interpretadas por el servidor antes enviadas al navegador del cliente. En aquellas partes donde el código *script* de servidor lo indica se añade la información que estipule, entremezclándose con el código *HTML* estático original. La Figura 1 muestra este esquema.

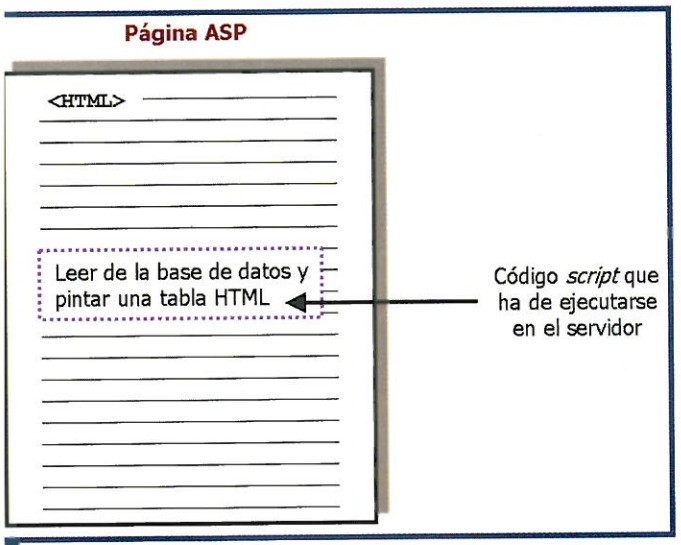


Figura 1.- Esquema de funcionamiento de las páginas ASP.

A continuación y a lo largo de esta serie aprenderemos a programar páginas *ASP*. Asimismo se analizará de qué manera se pueden integrar el *HTML Dinámico* y el lenguaje *XML*.

PREPARANDO EL ENTORNO DE TRABAJO

Las páginas *ASP* se ejecutan dentro de una arquitectura cliente/servidor como la que proporciona *Internet*. Sin embargo es posible reproducir este entorno en un ordenador personal con ayuda de un servidor. El único requisito es que este servidor esté preparado para reconocer e interpretar adecuadamente las páginas *ASP*. Entre las muchas posibilidades que ofrece *Microsoft* el servidor *PWS* (*Personal Web Server*) es una de las más sencillas. Este pequeño servidor es gratuito y viene con *Windows 98* o se puede instalar en *Windows 95* con un *Option Pack* que ofrece *Microsoft* también de manera gratuita a través de su servidor *Web*.

El *CD-ROM* de *Windows 98* incluye el servidor *PWS* aunque su instalación debe hacerse al margen de la instalación normal del sistema. En el directorio *\add-ons\pws* del *CD-ROM* encontraremos el archivo **setup.exe**. Ejecutándolo comienza la instalación del servidor, junto con el que también se instalarán los siguientes paquetes de *software*:

- Data Access Components: Este paquete está formado por: *ActiveX Data Objects* (*ADO*), *Remote Data Service* (*RDS*), *Microsoft OLE DB Provider for ODBC* y *Open Database Connectivity* (*ODBC*). Como ya se verá, estos componentes ofrecen la posibilidad de acceder a todo tipo de fuentes de datos que se encuentren en el servidor.
- *Easy administration:* Normalmente la administración de los servidores profesionales suele ser complicada, pero en el caso del servidor *PWS* se incluye un administrador cuyas funciones son muy fáciles de manejar.

EL ADMINISTRADOR DEL SERVIDOR

El administrador de *Web* es la aplicación que permite no sólo iniciar el servidor sino que además gracias a él podemos realizar la correspondencia entre las direcciones *URL* y los directorios reales de nuestro ordenador. La Figura 2 muestra la ventana del administrador de servidor. Una vez que se hace *clic* en el botón *Iniciar* el servidor se arranca y aparece un icono en el área de notificación de la barra de herramientas. De la misma forma la ventana principal del administrador cambia y presenta el aspecto que ilustra la Figura 3.

Junto con el servidor PWS se instalan los componentes de acceso a datos

La dirección *IP* de nuestro ordenador es **127.0.0.1** y podremos acceder a las páginas *Web* a través de la dirección *URL* **http://127.0.0.1/** o bien utilizando el nombre del equipo, por ejemplo, **http://dit6a6/**.

El siguiente paso consiste en realizar la correspondencia entre las directorios y las direcciones *URL* que reconoce el servidor, y que por lo tanto podremos visitar con nuestro navegador. Haciendo *clic* sobre el botón *Avanzada* podremos llevar a cabo esta tarea. La Figura 4 muestra cómo se realiza la asociación. Después de hacer *clic* en el botón **Aceptar** se establece en el servidor un vínculo entre la dirección **http://127.0.0.1/asp** y el directorio **C:\WEBSHARE\WWWROOT\asp** de manera que las páginas *ASP* contenidas en dicho directorio serán accesibles a

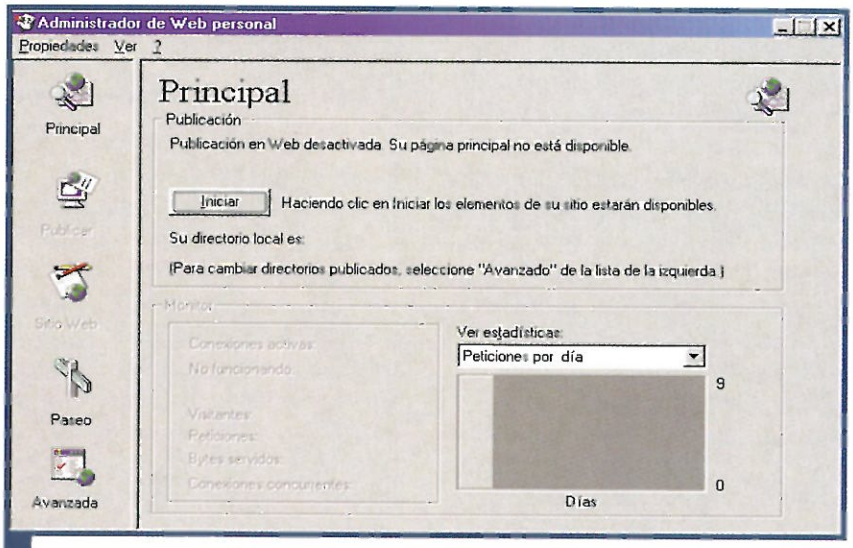


Figura 2.- Ventana de administración del servidor cuando éste está detenido.

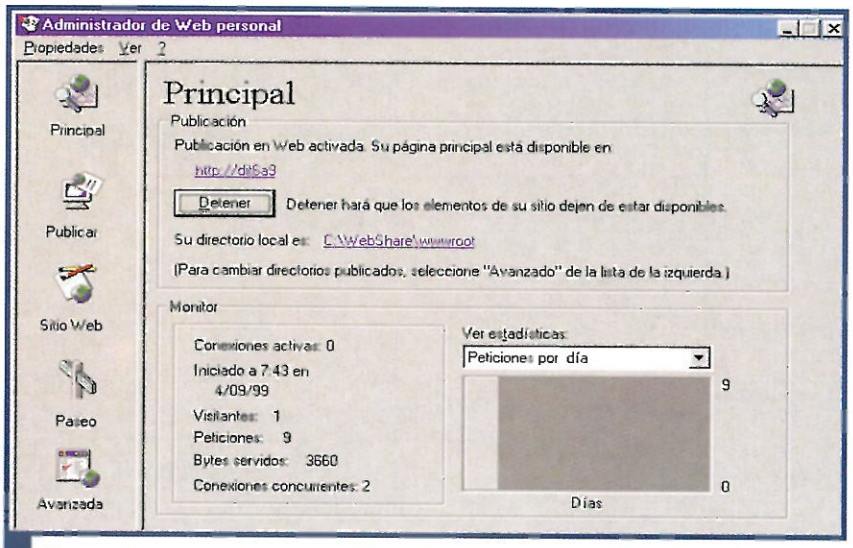


Figura 3.- Ventana de administración del servidor después de hacer clic en el botón Iniciar.

través de esa dirección *URL*. En este momento ya podemos empezar a programar.

Para editar páginas ASP basta un editor de texto simple

Las páginas *ASP* no son más que páginas *HTML* con código *script* de cliente añadido. Esto significa que para editarlas basta un editor de texto simple como *Notepad*. No obstante existen un gran número de editores especializados tanto en versiones de demostración de un producto comercial, como *shareware* o *freeware*.

Con respecto a este último grupo se recomienda visitar **http://www.nonags.com**, sitio *Web* donde se pueden obtener aplicaciones *freeware* que algunas veces no tienen nada que envidiar a las profesionales.

CARACTERÍSTICAS DE UNA PÁGINA ASP

Una página *ASP* no es más que una página *HTML* en la que además de las habituales etiquetas se encuentran otros elementos que son interpretados por el servidor. Estos elementos resultan ser secciones de código en *JScript* o *VBScript*. La diferencia principal con respecto a los *scripts* que aparecen con frecuencia en las páginas *Web* es que esas secciones se ejecutan en el servidor y no en el navegador del usuario.

DELIMITADORES

Para que el servidor reconozca las secciones de código que ha de ejecutar, antes de enviar la página al navegador del usuario deben aparecer los delimitadores que anuncian que lo que viene a continuación es código para el servidor. En las páginas *ASP* estos delimitadores son <% y %>. Así tenemos que la expresión:

```
<% var i = 1 %>
```

equivale a declarar en el servidor la variable *i* con un valor inicial igual a **1**.

Los delimitadores definen las secciones de código que ejecuta el servidor

Otro delimitador es aquel que indica una expresión de salida. Esto significa que el resultado de evaluar dicha expresión será puesto por el servidor en la página *Web* en ese mismo punto. La siguiente expresión:

```
<%= i %>
```

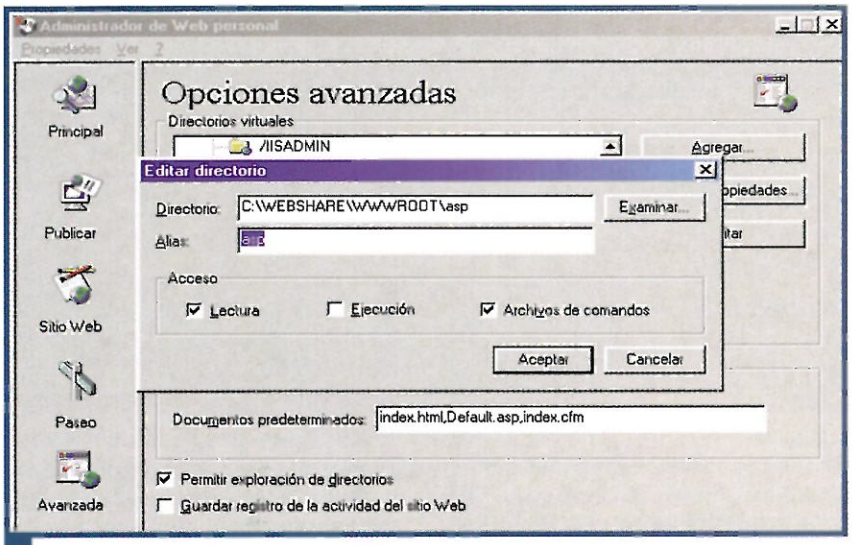


Figura 4.- Establecimiento de una correspondencia entre un directorio y una dirección URL.

es tanto como decir "pinta el valor de i en esta posición".

Estos son los elementos más importantes con los que podemos empezar a desarrollar páginas *ASP*. Ahora es el momento de ver cómo podemos crear "el relleno" que encierran esos delimitadores y qué representa en definitiva el grueso de la programación con páginas *ASP*.

LOS LENGUAJES DE SCRIPT: JSCRIPT Y VBSCRIPT

Las páginas ASP pueden ser programadas utilizando dos tipos de lenguajes de *script*: *JScript* y *VBScript*. Puesto que estos lenguajes se utilizan también para realizar *scripts* de cliente - que se ejecutan en el navegador -, es muy importante diferenciar claramente entre un caso y otro.

Los Listados 1 y 2, incluidos en el *CD*, contienen respectivamente el código fuente de una página *HTML* y una página *ASP* que realizan exactamente lo mismo. En ambos casos se calcula la fecha actual y se muestra en la página. Las figuras 5 y 6 ilustran el resultado.

Aunque las dos páginas hagan lo mismo, su modo de ejecución es completamente distinto. En el primer caso se trata de un *script* de cliente, es decir, de código *JavaScript* que se ejecuta en el navegador cuando la página llega al usuario. La página *ASP*, por el contrario, ejecuta su código en el servidor, y cuando llega al navegador del cliente el contenido no es más que *HTML*. Esto puede comprobarse observando el código fuente de la página del navegador. Mientras que en un caso el código de *script* sigue estando allí, en otro no hay nada ya que el servidor ejecuta el *script* correspondiente y simplemente traslada a la página aquellas expresiones que se encuentren entre delimitadores de salida.

A la hora de escoger entre *JScript* y *VBScript* son muchos los factores que pueden influir. Para esta serie de artículos se ha resuelto elegir *JScript* porque la experiencia que se adquiera utilizando este lenguaje servirá también para desarrollar *scripts* de cliente que se ejecuten igualmente en un navegador u otro. *VBScript* es un lenguaje fuertemente vinculado a *Visual Basic* y por ende a la plataforma *Microsoft* y su navegador *Internet Explorer*.

JScript es el mismo lenguaje que para los scripts de cliente

Por el contrario, aunque *JScript* tiene algunas características particulares, se trata básicamente del mismo *JavaScript* que es interpretado por *Netscape Communicator* e *Internet Explorer*. Eso aporta además una nueva ventaja: la mayoría de los desarrolladores de aplicaciones basadas en *Internet* tienen ciertos conocimientos de *JavaScript* – lo que no significa necesariamente que ocurra lo mismo con *VBScript* - y por lo tanto resultará mucho más fácil empezar a programar páginas *ASP* si el lenguaje utilizado es *JScript*.

Una vez más, esta elección no debe provocar confusión. Hubiera dado lo mismo el navegador utilizado a la hora de realizar los *scripts* de servidor de las páginas *ASP* ya que éstos son ejecutados en el servidor. Simplemente se ha optado por *JScript* ya que permite reciclar los conocimientos adquiridos para ser aplicados en *scripts* de cliente compatibles entre navegadores. Estos *scripts*, al ejecutarse en el navegador, sí que dependen de él.

OBJETOS PREDEFINIDOS

En todas las aplicaciones *Web* existen una serie de operaciones que se realizan con frecuencia, sean cuales sean las características

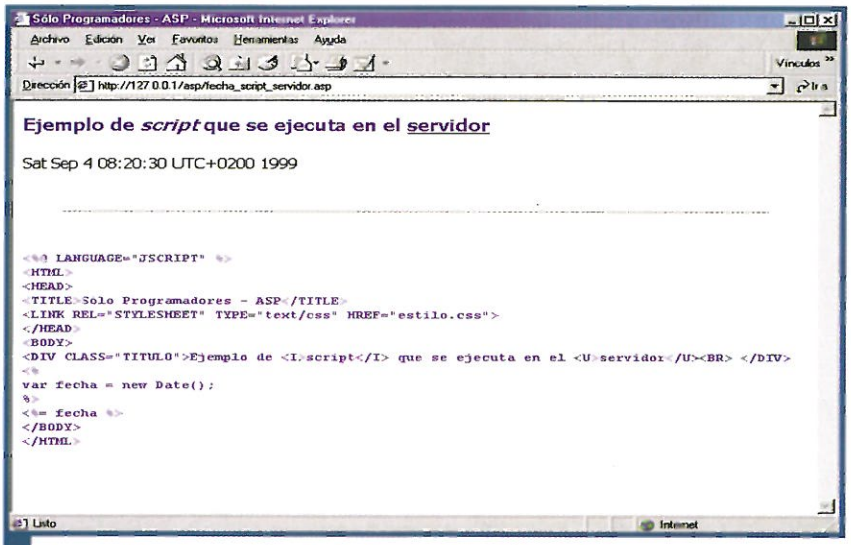


Figura 8.- Demostración del uso de la colección Form del objeto Request.

Tabla 2. Variables más importantes del servidor

Variable	Significado
AUTH_TYPE	Si la dirección URL tiene el acceso restringido se emplea esta variable para indicar el método de protección.
CONTENT_LENGTH	Tamaño de los datos que se mandan desde del navegador del cliente.
CONTENT_TYPE	Tipo MIME de los datos enviados.
GATEWAY_INTERFACE	Versión de la especificación CGI que cumple el servidor.
PATH_INFO	Esta variable contiene información con respecto a la ruta que aparece en la dirección URL. Sin embargo, la semántica de esta información viene dada por el programador.
Por ejemplo:	
http://www.sitio.com/mipagina.asp/dir/archivo?nombre=valor	
PATH_INFO =	/dir/archivo
QUERY_STRING =	nombre=valor
PATH_TRANSLATED	Valor de PATH_INFO una vez que el servidor ha realizado la traducción de direcciones lógicas a direcciones físicas.
QUERY_STRING	Esta variable contiene todo lo que figura a partir del signo de interrogación (?) en la dirección URL
REMOTE_ADDR	Dirección IP del agente que invocó a la página ASP. Esta no tiene por qué ser necesariamente la del cliente ya que si éste accede a través de un proxy, tomará el valor de la dirección IP del proxy.
REMOTE_HOST	Nombre completo del agente que invocó a la página ASP. Se comporta de la misma forma que REMOTE_ADDR
REQUEST_METHOD	Método HTTP empleado para enviar información al servidor: normalmente GET o POST.
SCRIPT_NAME	Dirección URL que identifica a la página ASP que ha sido invocada.
SERVER_NAME	Nombre del servidor tal y como se indicó en la parte host de la dirección URL de invocación de la página ASP. Puede ser una dirección IP o un alias de DNS.
SERVER_PORT	Puerto de atención del servidor.
SERVER_PROTOCOL	Versión y extensión del protocolo empleado por el servidor de información. Por ejemplo, HTTP/1.0
SERVER_SOFTWARE	Nombre y versión del software servidor que ejecuta la página ASP.

visibles en la página *Web*. Asimismo también es posible utilizar ciertos parámetros que afectan a la visualización misma de la página por el navegador, o para establecer *cookies*.

Las propiedades de este objeto son las siguientes:

- *Buffer:* Cuando esta propiedad toma el valor **true** significa que la página no será enviada al navegador del cliente hasta que no hayan terminado de ejecutarse todos los *scripts* contenidos en ella o hasta que los métodos *Flush* o *End* del objeto *Response* sean invocados. En caso contrario la página es enviada al cliente a medida que va siendo procesada.

  Esta propiedad sólo puede establecerse antes de que el servidor haya empezado a enviar la página por lo que de utilizarse siempre debe aparecer al principio de la página *ASP*.

La propiedad Buffer permite retrasar la respuesta del servidor

- *ContentType:* Esta propiedad sirve para fijar el tipo *MIME* de los datos que van a ser enviados al cliente. El valor por defecto es **text/HTML**, es decir, que el contenido por defecto será siempre una página *HTML*. Aunque se puede modificar. De hecho otros valores son utilizados para mandar otro tipo de datos, por ejemplo **image/JPEG** para enviar una imagen del tipo *JPEG*.
- *Expires:* Esta propiedad puede utilizarse para establecer la cantidad de tiempo que ha de transcurrir antes de que la página pase a considerarse como anti-

gua en la *caché* del navegador, la calificación dependerá del propio usuario.

- *ExpiresAbsolute:* La propiedad *ExpiresAbsolute* es similar a *Expires* aunque en este caso se fija la fecha y hora exactas en la que la página pasará a considerarse como antigua en la *caché* del navegador.
- *Status:* Sirve para fijar el mensaje de estado que devuelve el servidor. Mensajes de estado típicos son: *401 Unauthorized* o *310 Move Permanently*.

Los métodos del objeto *Response* son:

- *AddHeader:* Este método puede utilizarse para añadir cabeceras *HTTP* a las respuestas. No reemplaza ninguna de las cabeceras existentes y una vez que una cabecera ha sido escrita, no es posible eliminarla.

ContentType determina el tipo MIME de los datos

Una página *ASP* puede devolver cualquier tipo de información: texto simple, imágenes *GIF* o *JPEG*, código *HTML*, etc. Para que los navegadores sean capaces de mostrar esta información de forma correcta es necesario que conozcan de qué naturaleza es. Por ello, las cabeceras *HTTP*, especificadas por la interfaz *CGI*, proporcionan esa información. La cabecera más importante que conocemos hasta el momento es la que especifica el tipo *MIME* del contenido (*ContentType*).

Por otro lado están también lo que se denominan variables *HTTP*. Lo normal es definir una variable *HTTP* para cada una de las cabeceras que se reciben del cliente, que toma el nombre de *HTTP_(nombre_de_cabecera)*.

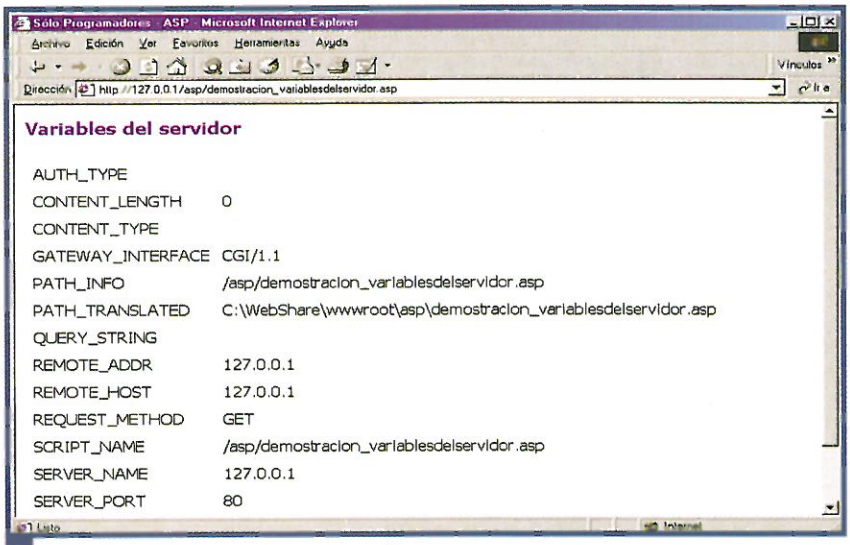


Figura 9.-Visualización de las variables del servidor.

- *AppendToLog:* Añade una línea en el archivo de *log* del servidor. Puede ser invocado tantas veces como se desee a lo largo de una página *ASP*. La cadena que recibe como parámetro no puede contener comas ni debe exceder de **80** caracteres.
- *BinaryWrite:* Este método es utilizado por el servidor cuando desea mandar datos binarios.
- *Clear:* El método *Clear* elimina el contenido que se encuentre almacenado en el *buffer* en el servidor. Si la propiedad *Buffer* no se ha establecido como **true**, la invocación del método *Clear* producirá un error.
- *End:* Cuando se invoca este método el servidor deja de procesar la página *ASP*.
- *Flush:* El método *Flush* fuerza a que se mande el contenido almacenado en el *buffer* hasta el momento. Si la propiedad *Buffer* no se ha establecido como **true**, la invocación del método *Clear* producirá un error.
- *Redirect:* Redirige el navegador del cliente hacia otra dirección *URL*. Por ejemplo, la siguiente expresión:

```
<%Response.Redirect("http://127.0.
    0.1/asp/demostracion_query_st
    ring2.asp?color=A7A789&esti-
    lo=negrita") %>
```

provocaría que el navegador del cliente fuese a esa dirección, con todos los parámetros propios de la *query string* incluidos.

Es posible llevar el navegador a otra dirección URL con el método Redirect

- *Write:* Sirve para escribir en la página que será visualizada por el navegador del cliente.

En este primer artículo hemos creado las bases para empezar a desarrollar páginas *ASP*. A lo largo de los siguientes profundizaremos en todos los temas que hasta el momento sólo hemos apuntado: los formularios, el acceso a fuentes de datos desde las páginas *ASP*, etc.

SOLARIS™

GRATIS 2º CD con una de las SUITES más prometedoras del momento

Instalación de StarOffice 5.1 para Solaris

Antonio José Novillo López

En estas páginas os explicamos cómo instalar *StarOffice* 5.1 que regalamos con la revista en este número, para el sistema operativo *Solaris* . Una suite 100 % funcional, que podréis utilizar sin límite de tiempo.

INTRODUCCIÓN

En este número os regalamos la versión completa de la Suite de Aplicaciones *StarOffice* 5.1 para la plataforma *Solaris* , tanto en su versión para *Sparc* como para *Intel x86*. Este conjunto de aplicaciones, desarrollado por la empresa alemana *StarDivision*, ha pasado a pertenecer a *Sun Microsystems*. Sin embargo, la política de distribución será muy similar a la mantenida anteriormente por la empresa alemana, por lo que todos los usuarios no comerciales podrán disfrutar de esta completísima herramienta.

Estos cambios convierten a *Solaris* en una alternativa más seria frente a *Windows 9x* y *Linux* como sistema operativo de sobremesa. A la tradicional y demostrada solidez de *Solaris* como sistema operativo, se une la flexibilidad de *StarOffice* como paquete integrado. Esto unido a la posibilidad de adquirir el sistema operativo por tan sólo 10$, en caso de uso no comercial, ha de suponer un fuerte relanzamiento de *Solaris*.

En menos de 20 minutos podremos ejecutar nuestras aplicaciones con StarOffice

Una de las mayores virtudes de *StarOffice* es su facilidad de instalación y manejo, así como el que todas las versiones para diferentes plataformas sean iguales. Esto nos lleva a poder instalar y trabajar con ella en plataformas tan diferentes como *Solaris* y *Windows 9x,* sin necesidad de aprender nuevos comandos. Además, en esta versión 5.1 se ha potenciado la integración de las distintas herramientas que forman el paquete entre sí y con el entorno en el que se instalan, haciendo su uso más intuitivo. Asimismo se ha mejorado la compatibilidad con otros formatos como los de *Microsoft*, de modo que podamos compartir nuestros ficheros.

TIPOS DE INSTALACIÓN

Podremos instalar el *software* de dos formas diferentes, para un único usuario o bien con una instalación multiusuario, también llamada de *red*.

INSTALACIÓN MONOUSUARIO

Para realizar este tipo de instalación debemos entrar en el sistema como un usario no *root*. Si realizamos la instalación con este usuario tendremos problemas y posiblemente el programa no nos deje finalizarla. Por tanto deberemos entrar en el sistema como un único usuario.

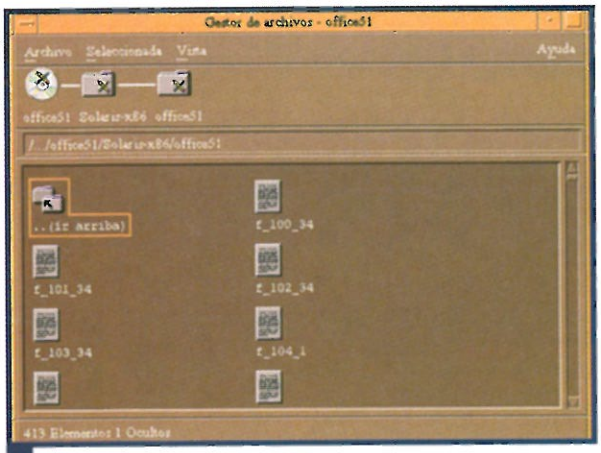


Figura 1. En los ficheros del *CD* encontraremos el programa *setup*.

El *CD-ROM* debe estar montado con anterioridad por el *root* si nosotros no tenemos permisos para montar un *CD*. Abriremos el *CD-ROM* desde nuestro Panel de Control. Encontraremos ante nosotros dos directorios, uno para *Sparc* y otro para *Intel x86*. Escogeremos la versión que nos interese y nos llevará a un directorio donde podremos encontrar un ejecutable denominado *setup*. Lo ejecutaremos, bien desde una consola o bien pulsando dos veces sobre él. En este último caso nos aparecerá una ventana indicando si queremos añadir algún parámetro. En nuestro caso no será así. Daremos a ejecutar y tras un rato, que variará según la potencia de nuestro procesador, nos aparecerá el logotipo de *StarDivision* junto con un mensaje que indica que la configuración se está iniciando.

Tras presentarnos la licencia de usuario y aceptarla, el sistema nos irá pidiendo la información del usuario, así como el directorio de instalación. Éste estará por defecto en el del usuario de instalación. En el caso de que queramos instalarlo en otro diferente, el usuario deberá poseer permisos de escritura en él.

Seguidamente deberemos escoger el tipo de instalación que deseamos realizar. Las tres posibles opciones son *Instalación Típica*, *Mínima* o *Personalizada*. Se recomienda realizar una instalación típica, dejando la personalizada para usuarios avanzados.

El proceso de instalación tardará alrededor de 20 minutos. El propio instalador crea un acceso a programa de iniciación de *StarOffice* en el panel de control del *CDE*.

INSTALACIÓN MULTIUSUARIO

En el caso de que queramos instalar el *software* en una estación de trabajo para varios usuarios, deberá ser el usuario *root* quien realice la instalación del paquete, y luego cada usuario tendrá que realizar una instalación personal.

Por tanto, cuando hayamos montado el *CD-ROM*, procederemos a escoger la versión apropiada para nuestro *hardware*, y lanzaremos el programa *setup*. En este caso deberemos añadir el parámetro */net* a la ejecución. En el caso de lanzarlo desde un terminal:

```
# setup /net
```

El proceso de instalación será muy similar al anterior. Pero el directorio en el que se encontrarán los ficheros estará en alguno de los directorios del sistema, como */usr* o */opt*.

Cuando hayamos finalizado el proceso, dispondremos de *StarOffice* en nuestro equipo. Ahora cada usuario deberá realizar una instalación personal en la que se copiarán algunos fichero, alrededor de 2 *Mb*, en su directorio.

Para ello iremos al directorio donde se haya instalado *StarOffice*, y ejecutaremos *setup*. Seguidamente se nos pedirán nuestros datos personales, el directorio donde queremos instalar el *software* y alguna opción más. Al igual que en caso monousuario, el icono de *StarOffice* se integrará en nuestro escritorio y sólo tendremos que pulsarlo para lanzar la aplicación.

EJECUCIÓN DE LA APLIACIÓN

Una vez haya finalizado la instalación podremos hacer uso del programa, **sin límite de tiempo**. Deberemos reiniciar el panel de control del gestor de ventanas para que nos aparezca el icono. Para lanzar el paquete sólo tendremos que pulsar sobre este icono o bien lanzar el fichero *StarOffice* que aparecerá en nuestro directorio.

La primera vez que lancemos el programa nos instará a que nos registremos. Esta opción no es obligatoria pero si recomendable. Teniendo en cuenta que es gratuita, podremos enviar nuestros datos bien por correo electrónico o bien por correo tradicional a *Sun*, donde apareceremos como usuarios registrados.

A partir de este momento podremos disfrutar de la potencia de esta Suite de Aplicaciones que cumplimenta unos de los puntos más flojos de *Solaris*, las aplicaciones ofimáticas. Desde ahora podremos editar nuestros ficheros de texto y salvarlos con formato *Word 2000 ó 97*, crear presentaciones y hojas de cálculo, así como leer nuestro correo o crear bases de datos personales.

CONCLUSIÓN

Como hemos podido observar, la instalación de *StarOffice* 5.1 es bastante sencilla y muy similar a la de cualquier programa de *Windows 9x*. Tan sólo debemos montar el *CD-ROM*, algo que por defecto hace solo el sistema desde el usuario *root*, y lanzar la apliación *setup*.

Así, siguiendo las instrucciones que nos aparecen en pantalla en menos de 20 minutos habremos instalado el *software* y estaremos en disposición de crear nuestros primeros documentos.

Criptografía (I)

Javier Sanz Alamillo. jsanza@teleline.es

La necesidad de disponer de medios de transmisión seguros y la criptografía se han convertido en temas cada vez más importantes. A lo largo de la serie de artículos que ahora iniciamos se mostrarán sus fundamentos, los métodos desarrollados y su aplicación utilizando Java.

La palabra *Criptografía* proviene del griego *kruptos,* oculto y *grafe,* escritura, por lo que la *criptografía* es el estudio de la escritura secreta, la forma de escribir ocultando el significado, una herramienta fundamental para la seguridad en los sistemas informáticos.

A continuación se van a mostrar sus fundamentos básicos y la aplicación de los métodos disponibles utilizando el lenguaje por excelencia para el desarrollo de aplicaciones en *Internet, Java*.

FUNDAMENTOS BÁSICOS

Antes de desarrollar los métodos criptográficos se van a presentar los fundamentos teóricos de la criptografía. Además, se van a definir algunos términos comunes que serán utilizados a lo largo de la serie.

PROPIEDADES EN LA SEGURIDAD DE LA INFORMACIÓN

Se suelen definir tres criterios o propiedades para la seguridad de la información, que son:

- *Confidencialidad* (el secreto): define que la información debe estar disponible sólo para los usuarios autorizados a manejarla. Esta es la propiedad más relacionada con los procedimientos criptográficos.

Los métodos criptográficos son esenciales para garantizar el uso muchas aplicaciones

- *Integridad*: se refiere a la garantía de que la información no puede ser alterada, modificada o manipulada sin que el usuario se dé cuenta de ello.
- *Accesibilidad*: determina quién accede a la información y en qué momento puede hacerlo; proporcionando los mecanismos para impedir el acceso de "entidades" no autorizadas para que las que lo están puedan disponer de ella.

Figura1.- Medios de transmisión poco seguros.

Existen otras características definidas que están más relacionadas con la seguridad en las redes de datos. En concreto:

- *Autenticidad*: Es la forma en la que se asegura el origen y destino de la información. Importante por ejemplo, para validar el correo electrónico.
- *Imposibilidad de rechazo o no repudio*: Permite asegurar que cualquier entidad que envía o recibe una determinada información no puede alegar que no la envió o recibió.

COMPONENTES DE UN CRIPTOSISTEMA

En un sistema criptográfico o criptosistema se pueden encontrar los cinco componentes siguientes:

- El *espacio de mensajes* $M = \{m_1, m_2, ..., m_n\}$: conjunto de todos los posibles textos en claro o textos que cifrar. Conjunto de mensajes inteligibles.
- *Espacio de textos cifrados o espacio de cifrados*, $\mathbf{C} = \{c_1, c_2..., c_n\}$: Conjunto de todos los mensajes cifrados.
- *Espacio de claves* $K = (k_1, k_2..., k_n\}$.
- Un conjunto de transformaciones de cifrado, E, tales que E_k: M -> C, donde k Œ $_k$ define una transformación.
- Un conjunto de transformaciones de descifrado, D, tales que D_k : C -> M, donde k Œ $_k$ define una transformación.

Todos estos elementos son necesarios para la correcta transformación de la información. Cuando a un mensaje m Œ M se le aplica una transformación E_k se obtiene un mensaje cifrado c Œ C. Para descifrar ese mensaje $m = D_k (E_k (m))$.

Por ejemplo, el método *Cesar* se basa en sustituir cada carácter del alfabeto por el que se encuentra en las tres posiciones siguientes. De tal forma que E consiste en sustituir una letra por la de más adelante, y que k, determina el parámetro, en este caso, tres posiciones.

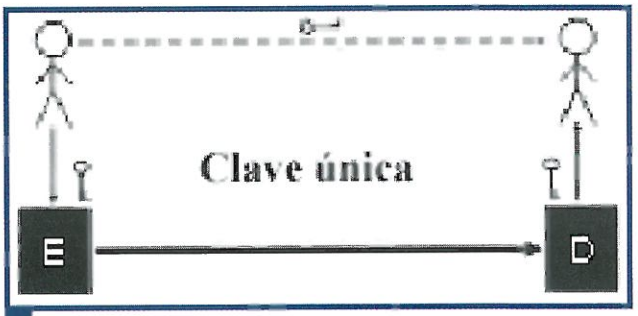


Figura 2.- Criptosistema de clave privada.

No siempre se utiliza la misma clave para el cifrado y el descifrado, por lo que en función del uso de la clave se distingue entre *criptosistemas de clave privada y pública*.

Realizar sustituciones en el mensaje a cifrar puede resultar, en ocasiones, fácil y eficaz

En un *criptosistema de clave única, simétrico o de clave privada* la fortaleza del cifrado reside en el secreto de la clave, pues la clave k se comparte en las transformaciones E_k y D_k.

Un *criptosistema asimétrico, de dos claves o de clave publica* es aquel en el que cada usuario U dispone de dos claves, *ub* o clave pública y *uv* o clave privada. Se basa en que es imposible obtener una a partir de la otra. La clave pública *ub* sirve para que cualquier usuario pueda enviar un mensaje a U usando una transformación E_{ub}, siendo las funciones E y D conocidas. Con ello se proporciona el secreto a los mensajes de U pero éste no puede averiguar quién cursó el mensaje puesto que E_{ub} es pública.

Para descifrar el mensaje se realiza la transformación D_{uv} conocida únicamente por U que posee la clave *uv*. Independientemente del tipo de criptosistema, todos, sin excepción, deben cumplir una serie de requisitos para poder ser utilizados en la práctica, como son:

- Las transformaciones de cifrado y descifrado deben ser eficientes para todas las claves.
- La seguridad del sistema debe depender del secreto de las claves y no del secreto de las funciones E y D.

BASES TEÓRICAS DE LA CRIPTOGRAFÍA

Las bases teóricas que permiten entender y utilizar los métodos criptográficos son las siguientes:

- La teoría de la información.
- Teoría de números.
- Teoría de la complejidad de algoritmos.

De estas tres, las bases elementales de la teoría de números se presentan en la aritmética modular.

ARITMÉTICA MODULAR

Define el concepto de congruencia:

Dado un entero positivo n y dos enteros a y b, se dice que a es congruente a b modulo n sí y sólo sí

```
a – b = k * n  para k entero.
```

n divide la diferencia $(a - b)$. Se expresa de la siguiente forma:

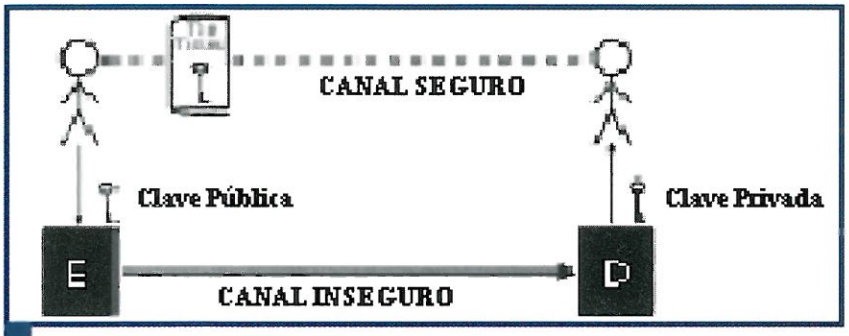


Figura 3.- Criptosistema de clave publica.

```
a ∫ b (mod n)
```

```
Por ejemplo, 12 ∫ 2 ( mod 5 ).
```

Si se cumple que *a ∫ b (mod n)* entonces el número *b* se llama *resto de a (mod n)* y recíprocamente *a* es un resto de *b (mod n)*, basta con recordar que *a = k * n + b*. Por tanto, dados dos números enteros *a* y *b*, se dice que son congruentes si el resto de dividirlos por *n* para ambos casos es el mismo.

Se define *el inverso multiplicativo o inverso* de un entero *a Œ {0, n-1}* al entero *x Œ {0, n-1}*, tal que *a*x (mod n)=1*. Por ejemplo, 5 y 3 son inversos *mod* 7 ya que 5*3 *(mod 7)*=1.

Dos enteros *a y b son primos* entre sí cuando su máximo común divisor es la unidad, representada por *m.c.d (a, b)=1.*

Existe el inverso multiplicativo, si se cumple la siguiente condición:

- Sea *a Œ {0, n-1}* y mcd(a,n)=1, o sea que *a* y *n* son primos entre sí, entonces existe un x *Œ {1, n-1} / a*x = 1 (mod n).*

Para calcular el *mcd(a,n)* se suele utilizar el antiquísimo algoritmo de *Euclides* que determina el siguiente proceso:

Sean *a* y *n* dos números de los que queremos calcular el *mcd*, aplicando la construcción de la tabla 1 se obtiene el *mcd*.

Inicialmente, se divide *n* entre *a* y el resto, r_1 se sitúa como se indica en la Tabla 1. Ese resto, r_1, es utilizado para dividir a c_1 ó cociente de dividir *n* entre *a*. Se repite este proceso hasta que obtengamos un resto igual a **0**. Si *a* y *n* son primos entre sí, entonces $r_n = 1$.

Gracias a la aritmética modular hoy se disponen de métodos criptográficos muy seguros

Por ejemplo, si se tuviera que calcular el *mcd (23,5)*, aplicando el algoritmo de *Euclides* se obtendría el resultado de la Tabla 2.

Por tanto, *mcd(23,5) = 1* y **23** y **5** son primos entre sí.

Se define *conjunto reducido de restos mod n* al conjunto de enteros $R = \{ r_1, r_2, ..., r_n \} \tilde{A} \{ 0,1, ... n-1 \}$ / mcd $(r_i,n) = 1$, es decir, son primos respecto de *n*. Así, el conjunto de restos *mod* 6 sería:

{1,2,3,4,5,}, eliminando los no primos con 6, R={1,5}.

Se define el *indicador de Euler de n (F(n))* como el número de elementos del conjunto reducido de restos módulo *n*. Por ejemplo, F(6)=2.

El cálculo de F(n) cumple las siguientes reglas:

- F(n)= n-1 si *n* es primo.
- $F(n^k) = n^k - n^{k-1}$, *n* primo.
- F(n.m)= (n-1)(m-1) con *n* y *m* primos.

Si *n* es cualquier número basta con descomponerlo en primos. El *teorema de Euler* demuestra que dados dos números enteros *a* y *n*, primos entre sí, se verifica que:

```
aF(n) (mod n) = 1
```

El *teorema de Fermat* demuestra que si *n* es un número primo:

```
a.x (mod n) = 1    y
x = aF(n) (mod n), entonces x =
    an-2 (mod n)
```

Por ejemplo, para resolver 3.x (mod 5)=1 se haría:

```
5 es primo, luego F(5)=4,
x = 33 (mod 5) = 2
```

Aunque se han presentado muchos conceptos sin demostración, e incluso algunos de ellos pueden parecer triviales, la mayoría de los sistemas de cifrado modernos, por no decir todos, están basados en ellos. Su aplicación se comprenderá en los siguientes métodos de cifrado y sobre todo en la descripción de los métodos criptográficos modernos.

Tabla 1. Fórmula para calcular el máximo común divisor (mcd).

	c1	C2	c3	...	Cn-1	cn	cn+1
N	a	R1	r2		rn-2	rn-1	rn=MCD
r1	r2	r3			rn	0	

Tabla 2. Ejemplo práctico del algoritmo de Euclides.

4	1	1	2	
23	5	3	2	1
3	2	1	0	

MÉTODOS CRIPTOGRÁFICOS CLÁSICOS

Constituyen el origen de la criptografía. Se basan en sustituciones y transposiciones de caracteres. Todos ellos se han estado aplicando hasta la aparición de ordenadores con cierta potencia de cálculo, por lo que se han creado métodos criptográficos mucho más complejos.

Además de los métodos clásicos propiamente dichos, se van a mostrar métodos de transformaciones matriciales y *booleanas*. Todos ellos son criptosistemas de clave privada.

SUSTITUCIÓN SIMPLE MONOALFABETO

Se basa en sustituir cada letra o carácter del mensaje (texto en claro) por otra letra que forma parte del cifrado (texto cifrado). Esta sustitución utiliza la siguiente transformación:

$$E(m) = (am \pm b) (mod\ n)$$

donde se tiene que:

- m, es el valor numérico asociado a cada letra del mensaje, estableciéndose según la siguiente correspondencia:

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*
1	2	3	4	5	6	7	8	9	10	11	12

m	*n*	*ñ*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*
13	14	15	16	17	18	19	20	21	22	23

w	*x*	*y*	*z*
24	25	26	27

- E es la función de cifrado,
- $E(m)$ es el valor asociado a la letra correspondiente del mensaje cifrado.
- b determina el desplazamiento entre las letras.
- a determina el intervalo de separación entre dos letras del cifrado cuando éstas son consecutivas.
- n es el número de letras del alfabeto.

Cuando la transformación utilizada define a=1 se denomina

¡Infórmate ya, plazas limitadas!

msdn foro de desarrolladores

A lo largo de un día completo, conocerás de primera mano la estrategia de Microsoft en el mercado de los desarrolladores y los recursos que Microsoft pone a tu disposición como desarrollador. Conocerás el hoy y descubrirás el mañana de los productos y tecnologías. Además, podrás elegir lo que realmente te interesa, seleccionando entre las diferentes sesiones prácticas con alto contenido técnico y eligiendo sólo aquellas que más te interesan a tí y a tu empresa.

FECHA	CÓDIGO	CIUDAD
5 octubre	MSDNFD03	Valencia
7 octubre	MSDNFD04	Madrid
18 octubre	MSDNFD05	Barcelona

Para información detallada de la agenda e inscripción al evento consulta
www.microsoft.com/spain/msdn/eventos/foro/
ó llama a nuestro teléfono de atención al cliente 902 197 198

msdn presentaciones GRATUITO

En una mañana trataremos en detalle, los temas técnicos de mayor actualidad. Son eventos gratuitos, impartidos por los mejores Ingenieros de Microsoft, exclusivamente para técnicos como tú. Además contaremos con la presencia de Compaq que contará su experiencia en la implantación de un caso real y de los centros de formación Microsoft CTEC Professional Training e Intec.
No puedes perderte este evento clave para el desarrollador.

MADRID

FECHA	CÓDIGO	ÁREAS
19 octubre	MSDN00P1	Comercio Electrónico
28 octubre	MSDN00P4	Windows DNA 2000
11 noviembre	MSDN00P5	Migración Windows 2000
23 noviembre	MSDN00P8	Datawarehousing
16 diciembre	MSDN00P10	Gestión de conocimiento

BARCELONA

FECHA	CÓDIGO	ÁREAS
21 octubre	MSDN00P2	Comercio Electrónico
26 octubre	MSDN00P3	Windows DNA 2000
16 noviembre	MSDN00P6	Migración Windows 2000
18 noviembre	MSDN00P7	Datawarehousing
14 diciembre	MSDN00P9	Gestión de conocimiento

Para información detallada de la agenda e inscripción al evento consulta
www.microsoft.com/spain/msdn/eventos/presentaciones/
ó llama a nuestro teléfono de atención al cliente 902 197 198

método del tipo *César*. Así por ejemplo, con $b=3$, el texto en claro "solo programadores" se convertiría en "vrñr surjudodgruhv".

Veamos la implementación del método *César* utilizando *Java*, para cualquier valor de b, ya que sabemos que a valdrá **1**. Se deja al lector la modificación de este algoritmo para valores de a diferentes de **1**.

La implementación de este método sólo requiere ir utilizando las letras del abecedario e ir cambiando la letra de una determinada posición origen a otra desplazada, configurando así el texto cifrado.

Para ello, el cifrado simplemente consiste en cifrar la cadena carácter por carácter, determinar el valor de su posición en el abecedario, añadir a este valor el de b o desplazamiento, obtener este valor del mismo alfabeto y situarlo como parte del mensaje cifrado.

El método César es muy fácil de implementar

Este proceso queda perfectamente definido en el método cifrar. En el proceso contrario, descifrar, únicamente hay que tener en cuenta que el desplazamiento ya no va a ser el valor de b, porque estaríamos volviendo a desplazar hacia "delante", sino que debemos aplicar un desplazamiento contrario, por lo que se hace que b valga el número de caracteres del alfabeto menos el desplazamiento. En el Listado 1, se puede observar una sencilla implementación del método *César*.

El carácter blanco no forma parte del abecedario. Pruebe a realizar las modificaciones adecuadas para su utilización y observe qué resultados se pueden obtener.

Tabla 2. estructura TVIDEOHDR

Campo	Descripción
lpData : pBYTE;	*Buffer* que contiene los datos de la imagen
dwBufferLength : DWORD	Longitud del *buffer*
dwBytesUsed : DWORD	N° de *bytes* con información dentro del *buffer*
dwTimeCaptured : DWORD	Milisegundos de captura que llevamos
dwUser : DWORD	Reservado
dwFlags : DWORD	Reservado
dwReserved : array[0..3] of DWORD	Reservado

Este método posee una serie de inconvenientes. Pruébese a cifrar una serie de caracteres como son "abcdefgh" y observe el cifrado resultante. Tras un detenido estudio, observará rápidamente cuál es el desplazamiento aplicado.

Es más, ¿qué ocurre si el texto que se va a cifrar son letras idénticas, del estilo "aaaaa"?; se obtiene el mismo resultado desplazado, por lo que no sólo podemos averiguar b sino incluso el a aplicado.

Para evitar estos ataques, se puede utilizar una equivalencia aleatoria entre los caracteres del alfabeto y su correspondencia en el cifrado, de tal forma que se podría tener que:

```
A -> D, B -> C , C -> Z, D -
> E ... Z -> F
```

Aunque todos los métodos de esta categoría se basan en el uso de varias sustituciones simples en el cifrado de un texto en claro, se pueden incluir modificaciones.

1) Viginére

En este método se utiliza una palabra clave cuyas letras definen los desplazamientos con las letras del mensaje original. Así por ejemplo, se tendría que el texto en claro "los cielos" aplicando como clave la palabra "clave" se obtendría "nzsx-mevooñ". La transformación queda definida mediante:

```
E(mj) = (mj + ki) mod 27
```

donde k_i representa el desplazamiento del alfabeto, m_j es la letra del mensaje en la posición j y $E(m_j)$ es la letra cifrada.

2) Método de autoclave

Se comienza a cifrar con la clave, pero se continúa cifrando con el propio texto en claro. Por tanto si el texto en claro fuera "holaamigos", la clave estaría compuesta por una palabra, por ejemplo, "solop" y concatenando a ésta la secuencia del texto en claro hasta que ambas se igualen en longitud. La clave definitiva a aplicar sería "solopholaa".

SUSTITUCIÓN HOMOFÓNICA

Aquí se establece una aplicación entre el conjunto de caracteres del alfabeto a utilizar A_M y el conjunto potencia de los elementos del alfabeto del cifrado A_C, de forma que:

$$E: A_M \rightarrow 2^{Ac}$$

2^{Ac} muestra el conjunto de posibles subconjuntos que se pueden formar con elementos de A_C, a los elementos de cada subconjunto en que se puede cifrar un solo carácter se les denomina homófonos. Este método lo que trata es de destruir la frecuencia de aparición de los caracteres del mensaje.

Veamos un ejemplo. Supongamos que disponemos de un alfabeto

LISTADO 1. Ejemplo de implementación del método *César*.

```
import java.io.*;
class cesar
{
    public static int long_alfabeto = 27 ;
    String mensaje ;
    int    desplazamiento ;
    int    desplazamiento_descifrado;
    int    long_mensaje;
    String alfabeto = "abcdefghijklmnñopqrstuvwxyz" ;

    cesar ( String cadena, int desp )
    {
        mensaje = cadena.toLowerCase();
        desplazamiento = desp ;
        desplazamiento_descifrado = long_alfabeto - desplazamiento;
        long_mensaje = cadena.length();
    }

    public String cifrar ( )
    {
        int pos_claro = 0 ;
        int pos_cifrado = 0;
        StringBuffer mensaje_cifrado = new StringBuffer("") ;
        char car_aux = 0;

      for ( int cont_car = 0 ; cont_car < long_mensaje ; cont_car ++
)
      {
            car_aux = mensaje.charAt ( cont_car );
pos_claro = alfabeto.indexOf ( car_aux );              pos_cifrado = (
Math.abs (pos_claro + desplazamiento )  % long_alfabeto );
        mensaje_cifrado.append ( ""+ alfabeto.charAt ( pos_cifrado ));
        }

        mensaje = mensaje_cifrado.toString() ;
        return mensaje;
    }

    public String descifrar ( )
    {
        int pos_claro = 0 ;
        int pos_cifrado = 0;
        StringBuffer mensaje_descifrado = new StringBuffer("") ;
        char car_aux = 0;

       for (int cont_car = 0 ; cont_car < long_mensaje ; cont_car ++
)
        {
            car_aux = mensaje.charAt ( cont_car );
            pos_claro = alfabeto.indexOf ( car_aux );
pos_cifrado = ( (Math.abs(pos_claro +   desplazamiento_descifrado )) %
long_alfabeto );
mensaje_descifrado.append ( ""+ alfabeto.charAt ( pos_cifrado ) );
        }

        mensaje = mensaje_descifrado.toString() ;
        return mensaje;
    }

    public static void main ( String args[] ) throws Exception
    {
        String mensaje_claro, mensaje_cifrado ;

        DataInputStream di = new DataInputStream (System.in) ;
        mensaje_claro = di.readLine();

        cesar mc = new cesar ( mensaje_claro, 3 );

        mensaje_cifrado = mc.cifrar();
       System.out.println ( "Mensaje cifrado : " + mensaje_cifrado );

        mensaje_claro = mc.descifrar();
      System.out.println ( "Mensaje descifrado : " + mensaje_claro );

        System.in.read();
    }
}
```

de 11 caracteres y deseamos construir con ellos mensajes cifrados mediante este sistema. Para ello, definimos unos valores numéricos que haremos corresponder con la aparición repetida de cada carácter. En el Listado 2, ubicado en el *CD-ROM*, se puede ver que *a* como primera letra del alfabeto le corresponde un 3 la primera vez que aparezca, 4 la segunda, y 6 la tercera, repitiéndose este sistema si el carácter volviera a aparecer.

La sustitución homofónica es un método algo más evolucionado

La asignación de valores se realiza aleatoriamente, aunque se deberían tener en cuenta determinadas circunstancias como la frecuencia de aparición de los caracteres.

Por ejemplo, si tuviéramos el texto en claro siguiente y utilizamos los valores numéricos definidos en el código *Java* que se detalla a continuación, tendríamos:

```
solo          programadores
10,0,15,5
     13,14,7,19,14,3,17,4,18,9,14,
     1,11
```

Una vez que se han asignado los valores para cada carácter del alfabeto únicamente queda definir el proceso de cifrado. Básicamente el proceso se basa en almacenar cuántas veces ha ido apareciendo un determinado carácter, localizar qué carácter numérico se le va a asignar y repetir el proceso hasta finalizar el conjunto de caracteres. Un estudio detenido del código fuente hará comprender lo sencillo que es este método y lo fácilmente implementable que es gracias a *Java*.

Lo interesante de los métodos de sustitución homofónica es que

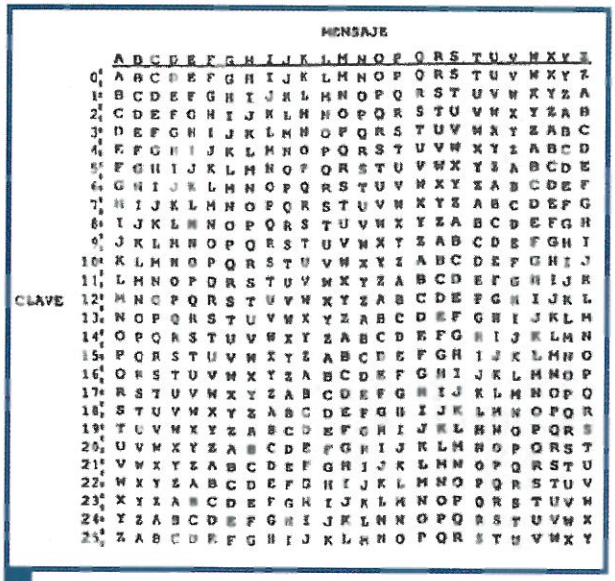


Figura 4.- Tabla Viginére.

son más difíciles de resolver que los de sustitución monoalfabética si se consigue destruir la frecuencia de aparición de los caracteres.

SUSTITUCIÓN DE TRANSPOSICIÓN

Con este método no se realiza ninguna sustitución de caracteres, sino que se altera su posición dentro del mensaje cifrado. El inconveniente es que no se altera la frecuencia de aparición característica del texto. De hecho existen varios métodos para alterar el orden:

1) Permutación por grupos.
Se trata de dividir el mensaje en grupos de un determinado tamaño y aplicar sobre ellos una alteración de orden.

Así por ejemplo, se puede definir el tamaño de grupo de cuatro elementos y para cada carácter M={ m_1, m_2, m_3, m_4} definir su nueva posición tal que:

```
Posición i   =  1 2 3 4
      f(i)   =  3 4 2 1
```

Esto simplemente viene a indicar que el primer carácter del grupo aparecerá en la tercera posición de ese grupo dentro del cifrado, el carácter en segunda posición aparecerá el cuatro y así sucesivamente. Así, si tuviéramos el grupo H O L A como texto en claro, aparecería L A O H como texto cifrado.

En el Listado 3, incluido en el *CD*, se observa la implementación de este método. Se ha escogido como tamaño de grupo cuatro elementos y la función de localización de los caracteres en el cifrado se ha definido aleatoriamente.

Los métodos de transposición alteran la posición de las letras

Como se han definido grupos de cuatro elementos, se deben añadir caracteres en blanco para rellenar este grupo y no producir así algunos "efectos laterales" en el programa. Ya que se realiza un relleno, en caso de un ataque al texto cifrado, esto podría dar información sobre el método de cifrar utilizado.

2) Posicionamiento en zig-zag.
Divide el mensaje en filas de caracteres colocándolo en zig-zag. Por ejemplo, si se tuviera el mensaje en claro:

```
SOLO PROGRAMADORES
```

Por lo que el mensaje cifrado resultante sería:

```
SLPORMDRSOORGAAOE
```

En el Listado 4, también incluido en el *CD-ROM*, aparece la implementación de este método.

3) Figuras geométricas.
Define un figura a seguir a la hora de crear el mensaje cifrado. Si tuviéramos por ejemplo el texto en claro **SOLO PROGRAMADORES** y se definiese un grupo de cinco letras por filas, se tendría que desarrollar la siguiente figura para construir el texto cifrado:

Aplicando el camino de la figura al mensaje:

```
S O L O P
R O G R A
M A D O R
E S Z Z Z
```

Y utilizando el carácter de relleno Z, el texto cifrado sería:

```
S R M E O O A S L G D Z O R
O Z P A R Z
```

En el Listado 5, en el *CD-ROM*, se encuentra el desarrollo *Java*. Analice el código y practique implementando su descifrado. Comprobará como la programación ya no es tan simple e inmediata, aunque con este método se pueden conseguir mejores resultados.

SUSTITUCIÓN POLIGRÁFICA

Estos métodos se basan en sustituir conjuntos de diagramas, trigramas, etc., de letras en vez de caracteres simples. El objeto es romper la frecuencia de aparición de determinadas letras propias del lenguaje. El método más común de este grupo de sustitución es el *Hill*.

El método de cifrado *Hill* se basa en cifrar usando k caracteres del texto en claro mediante la utilización de una sencilla transformación basada en una matriz A que realiza la tarea de clave.

En el caso que se optara por realizar la sustitución por diagramas, se tendría lo siguiente:

```
k=2,  M = m1 m2, y una
matriz A dim(k,k)

c1 = ( a11*m1 + a12*m2 ) mod
n
c2 = ( a21*m1 + a22*m2 ) mod
n
```

Si se definen las siguientes características de cifrado, se obtendría:

```
   | 7      6 |
A= | 3     11 |, con k = 2, n= 27
   ,

Texto en claro : so lo pr og ra
    ma do re
Texto cifrado  : hg fj ed gd ra
    dj dm oq
```

Todo este proceso puede resultar engorroso, pero su programación hace que resulte muy sencillo de utilizar. El Listado 6, que se puede encontrar en el *CD-ROM*, implementa el método *Hill* para k=2. El método de cifrado básicamente realiza dos llamadas para realizar los cálculos c_1 y c_2 y a partir de ahí realizar las sustituciones de caracteres como se han venido haciendo hasta ahora.

Este método trata de romper la frecuencia de los caracteres

Quedaría pendiente definir el proceso de descifrado. Pero hay algo más importante. ¿Cómo se llevaría a cabo?. La cuestión se resume en que la matriz que actúa como clave y en la que se basan las sustituciones A, debe tener inversa, por lo que se debe cumplir que:

```
A. A-1 mod n = I, donde I
es la matriz identidad.
```

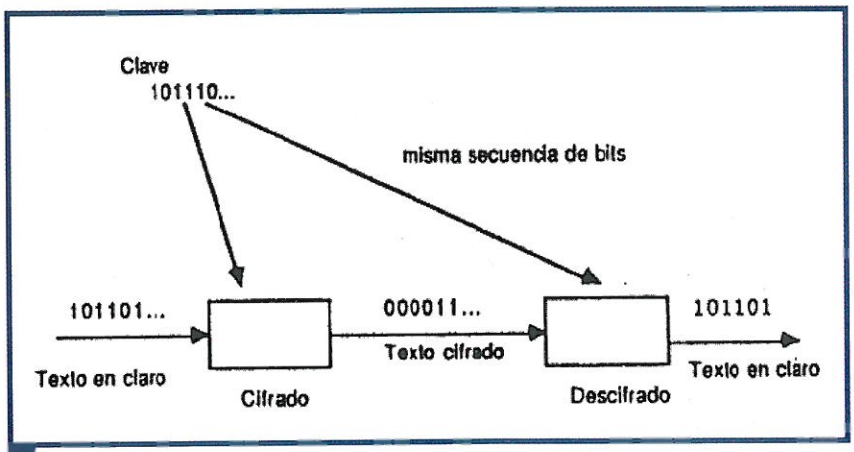


Figura 6.- Método Vernam

TRANSFORMACIONES LÓGICAS BOOLEANAS

Se pueden utilizar operaciones booleanas en las transformaciones. Pero de todas las operaciones booleanas posibles únicamente se pueden utilizar la negación, la equivalencia y el O-exclusivo o suma lógica puesto que son los que poseen operaciones inversas.

Dado un mensaje **M** en binario y una clave **K** de la misma longitud, el texto cifrado **C** puede ser, utilizado "O exclusivo":

```
Ek(M) = M ≈ K
```

Por lo que también se cumple lo siguiente:

```
M = Ek(M) ≈ K
```

Un cifrador que utiliza este método es el denominado *Vernam*, que utiliza como clave una secuencia aleatoria de números de igual tamaño que el texto en claro.

En el Listado 6, incluido en el *CD-ROM*, se muestra la aplicación de la lógica booleana. Su desarrollo es bastante simple, básicamente se basa en ir carácter a carácter aplicando el *O* exclusivo con un valor numérico que actúa como clave.

Como se ha podido comprobar, los métodos clásicos suelen ser bastante fáciles de comprender y aplicar, por lo que su aplicación informática es relativamente sencilla y práctica.

En próximos artículos se tratarán los métodos criptográficos modernos, tanto los criptosistemas de clave privada, como el famoso *DES* hasta los más complicados de comprender, utilizar y romper: los criptosistemas de clave privada, como el archiconocido *RSA*.

CONCLUSIÓN

Como se ha podido comprobar a lo largo de este artículo, los métodos criptográficos clásicos se basan en sencillas transformaciones que son fáciles de recordar y aplicar, por lo que son ideales para iniciarse en el interesante mundo de la criptografía, aunque actualmente su aplicación no es recomendable, porque implica asumir ciertos riesgos.

De todos modos, el lector ha podido observar cómo han evolucionado, desde el sencillo *César* hasta otros más desarrollados y evolucionados como el *Hill* o *Vernam*.

Captura de secuencias con Delphi (y IV)

Juan Luis Ceada (i5939@fie.us.es)

Para finalizar esta serie desarrollaremos dos pequeñas aplicaciones. Aunque sólo estaba previsto crear una pequeña aplicación de videoconferencia, al final se ha optado por diseñar también una aplicación que permitirá detectar movimiento.

INTRODUCCIÓN

En las entregas anteriores hemos creado un componente que permite capturar/visualizar vídeo de una forma más o menos cómoda. Ahora se van a desarrollar dos pequeñas aplicaciones para demostrar algunas de sus posibilidades. La primera es una aplicación de videoconferencia, mientras que la segunda actuará como un detector de movimiento.

Básicamente, el ordenador es capaz de saber cuándo hay movimiento en la escena que capta la cámara. El detector de movimiento funcionará sin problemas en *Delphi* 3 y 4. La aplicación de videoconferencia, al usar los componentes *Fastnet Tools for Delphi 4*, de la empresa *NetMasters L.L.C*, sólo podrá ser compilada en *Delphi* 4. De todas formas, es posible bajar de la página *web* de la empresa los componentes para *Delphi* 3, con lo que, con mínimos cambios, la aplicación debería funcionar también en *Delphi* 3. En el *CD-ROM* que acompaña a la revista, además de incluir las fuentes, como suele ser habitual, se incluyen los ejecutables de las dos aplicaciones. De esta forma, cualquier persona podrá comprobar los programas.

LA VIDEOCONFERENCIA

Esta aplicación permitirá enviar vídeo y audio (de una forma un tanto burda, eso sí) a través de una red *TCP/IP*. Puesto que no es motivo de este artículo hacer un estudio de esta familia de protocolos, usaremos unos componentes que permitirán olvidarnos de las comunicaciones, reduciendo el código dedicado a este aspecto a unas pocas líneas.

Dichos componentes se pueden obtener en **www.netmastersllc.com**, y aunque no proporcionan las fuentes, no presentan ninguna limitación de uso. He podido observar que al instalar los *Service Pack* 1 y 2 para *Delphi* 4, se actualizan los componentes de *Internet*, añadiendo los que vamos a usar en la aplicación.

Para las comunicaciones, usaremos dos componentes: TNMStrmServ y TNMStrm

Para poder probar la aplicación necesitaremos tener instalado en nuestra máquina el protocolo *TCP/IP*. Para las pruebas locales, bastará con instalar el protocolo. Pero para las pruebas en *Red*, tendremos que asignar a nuestro equipo una dirección *IP* válida.

IMPLEMENTACIÓN

En este caso, la aplicación se divide en dos partes: el cliente y el

servidor. El cliente incluye un componente *TImage* donde se mostrarán las imágenes recibidas, dos *TEdit* (que permitirán cambiar los puertos por donde se recibirán los datos), un *TButton*, para actualizar el valor del puerto y dos componentes *TNMStrmServ* encargados de recibir el audio y el vídeo.

La aplicación del servidor incluye un componente *TJLCVideoPanelNew*, un *TImage* (donde se muestran los *frames* que se envían), dos *TEdit* para indicar la dirección *IP* y el puerto de la máquina que aloja a la aplicación cliente. De igual forma existen algunos controles y botones auxiliares (para activar el vídeo, iniciar la transmisión, modificar el número de *F.P.S.*, y la calidad de las imágenes enviadas, etc.). Además, incluye dos componentes *TNMStrm* encargados de enviar el audio y el vídeo.

Todos los datos se envían con una llamada al método PostIt y se reciben con el evento OnMSG

Para las comunicaciones usaremos dos componentes: *TNMStrmServ* y *TNMStrm*. Los dos se usan para enviar *streams* de cualquier tipo a través de la *Red*, actuando el primero como servidor (recibe los *streams*) y el segundo como cliente (envía los *streams*). Para nuestro propósito, lo más conveniente sería que el servidor enviase los datos, y que los clientes los recibiesen. Por lo tanto, en la aplicación servidora (la que recibe las imágenes de la antena y las envía por la *Red*), incluiremos un componente *TNMStrm* (cliente), al que daremos el nombre de **Servidor**, y en la aplicación cliente, incluiremos un componente *TNMStrmServ* (servidor), al que llamaremos **Cliente**. El único problema que se presenta al hacer este cambio, es que la aplicación servidora debe conocer la dirección *IP* del cliente, cuando lo lógico sería que fuesen los clientes los que conociesen la dirección del servidor, y estableciesen comunicación con él.

El funcionamiento de estos componentes es muy simple. En el caso de *TNMStrm*, tendremos que indicar el puerto y la dirección *IP* de la máquina donde se encuentra el componente servidor. En el caso de *TNMStrmServ*, sólo tendremos que indicar el puerto en el que el servidor atenderá a las peticiones que le hagan los clientes. El envío de datos se realiza mediante una llamada al método *PostIt*, mientras que la recepción se realiza asignando código al evento *OnMSG* que proporciona *TNMStrmServ*.

ENVIANDO/ RECIBIENDO IMÁGENES

El código que se encarga de enviar/recibir una imagen por la *Red* se muestra en el Listado 1, en el *CD*. Como se puede observar se utiliza la función *SetDIBits*, que ya vimos en el artículo anterior.

La transmisión de imágenes da comienzo cuando el usuario pulsa el botón **Iniciar TX**. En ese momento, se llama al método *GrabarVideoSinDisco*, que simula una captura de vídeo. Las operaciones que se realizan son las mismas, pero el resultado no se almacena en disco.

Al iniciarse la grabación, cada vez que se capture un *frame*, se ejecutará el código asociado al evento *OnVideoStream*. En este caso, se copia el *frame* al *TImage*, y a continuación, se graba el *bitmap* en formato *JPG*, con una calidad determinada. Este archivo *JPG* es usado para rellenar la variable **MiStream** (de tipo *TFileStream*), que es la que se envía a través de la *Red*. El hecho de guardar la imagen en formato *JPG* implica dos cosas: es necesario incluir la unidad *JPEG* en la cláusula *uses*, y reducimos el volumen de datos enviados por la *Red* (de unos 50*Kb* se pasa a 3-5*Kb*, con una calidad del 40%).

La recepción consiste únicamente en copiar el *stream* recibido en un fichero, y posteriormente, cargar ese fichero en el *TImage*.

ENVIANDO/ RECIBIENDO AUDIO

Veamos ahora cómo enviar sonidos a través de la *Red*. Básicamente se trata del mismo procedimiento que hemos usado para transmitir imágenes. Crearemos un fichero *WAV*, que será el que enviemos por la *Red*. Ya vimos el mes pasado cómo tomar una muestra de sonido y convertirla en un fichero *WAV*. En este caso, la parte fija de la cabecera se crea al pulsar el

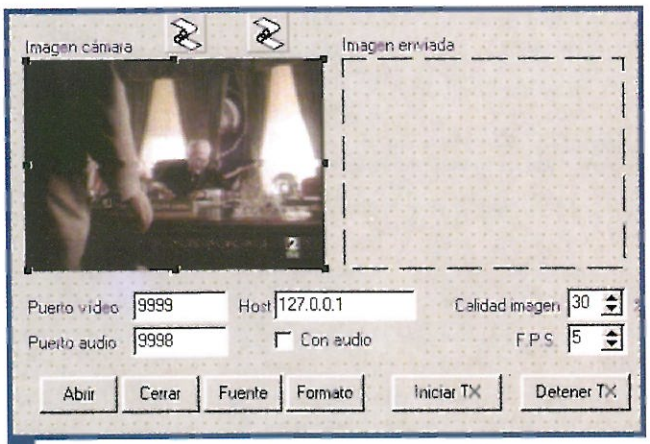


Figura 1.- Implementando la aplicación servidora. El componente está activo en tiempo de diseño.

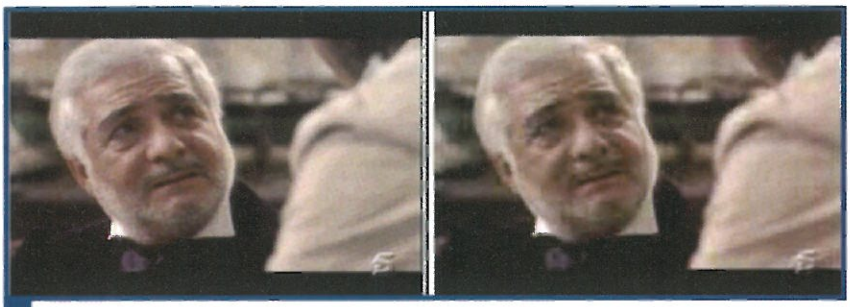


Figura 2.- A la izquierda se puede observar la imagen que captura la cámara, a la derecha la que se envía por Red. Calidad versus Tamaño.

botón **Iniciar TX**. Después, en el evento *OnWaveStream*, la completamos, y la unimos a los datos, para formar el fichero *WAV* (como se puede observar en el Listado 2, incluido en el *CD-ROM*). Más tarde, el cliente recibe el fichero, y lo reproduce.

Lamentablemente, los ficheros *WAV* ocupan bastante, por lo que deberíamos limitar la captura de audio a *8Khz*, 8 *bits* y monoaural (estos dos últimos son valores por defecto, por lo que no se modifican en la aplicación). También vimos en la entrega anterior cómo podíamos variar la calidad del audio capturado, mediante las funciones *capGetAudio-Format* y *capSetAudioFor-mat*. Todo esto se realiza al pulsar sobre el botón **Iniciar TX.** Durante las pruebas se ha podido comprobar como el envío de audio por la *Red* no proporciona la misma calidad que cuando se envían imágenes (se producen cortes en la reproducción del sonido).

PRUEBAS

Una vez ejecutadas las dos aplicaciones, en la servidora procederemos a conectar el vídeo (botón **Abrir**). Después seleccionaremos la fuente de la señal de entrada, y el formato de imagen de los *frames* capturados. Conviene elegir un tamaño de imagen de 192x144 por dos razones: los controles *TImage* de las aplicaciones tienen ese tamaño, y así reduciremos el volumen de datos a enviar. Podemos cambiar también la calidad de las imágenes (un valor del 30% es aceptable), así como el número de *F.P.S*. En una *Red* local, de 10*Mb/s*, con dos ordenadores conectados (un *Pentium* II y un 486), he podido enviar 10 imágenes por segundo, sin que ninguno de los dos equipos se haya resentido. También es posible, si se desea, activar la captura de audio.

Para las pruebas en una misma máquina conviene poner en silencio la entrada externa de la tarjeta de sonido

Por defecto, las dos aplicaciones están configuradas para trabajar en la misma máquina (*IP* 127.0.0.1). Si desea hacer las pruebas en una *Red* local, ejecute la aplicación cliente en una de las máquinas (preferiblemente en modo local, cópiela si es necesario al disco duro del ordenador). A continuación, ejecute la aplicación servidora en un equipo dotado de sintonizadora, indique la *IP* de la máquina donde está el cliente e inicie la transmisión.

Con respecto al audio, comentar que el volumen de la grabación se corresponde con el que tengamos asignado a la entrada externa de nuestra tarjeta de sonido (podemos cambiar este valor mediante el mezclador de *Windows*). Una vez ajustado el volumen, podemos poner en silencio dicha entrada, con lo que sólo escucharemos el sonido que se recibe por la *Red*. De otra forma, escucharíamos a la vez el sonido procedente de la sintonizadora y el de la *Red*, formándose un galimatías que puede volver loco a cualquiera.

EL DETECTOR DE MOVIMIENTO

Para que esta aplicación resulte sencilla supondremos que disponemos de una cámara, situada fija en una posición determinada y que enfoca a una zona con iluminación más o menos constante. Por ejemplo, una cámara situada en una esquina, dentro de una habitación iluminada artificialmente. La detección de movimientos es un tema que se puede complicar bastante si tenemos que tener en cuenta los cambios de luz, el movimiento de objetos debido al viento, etc.

Esto no quiere decir que no se pueda trabajar con cámaras situadas en zonas exteriores, pero lo que es seguro es que todo se complica demasiado. Una vez hemos dejado claro este aspecto, pasaremos a desarrollar la aplicación.

DIFERENCIA DE IMÁGENES

Para detectar movimiento, realizaremos una simple diferencia de imágenes. Nos basaremos en lo siguiente: tenemos una imagen base, que podría ser una toma de la zona que enfoca la cámara en estado "normal". Posteriormente, iremos capturando imágenes a intervalos regulares. Estas imágenes, las iremos restando de la imagen base.

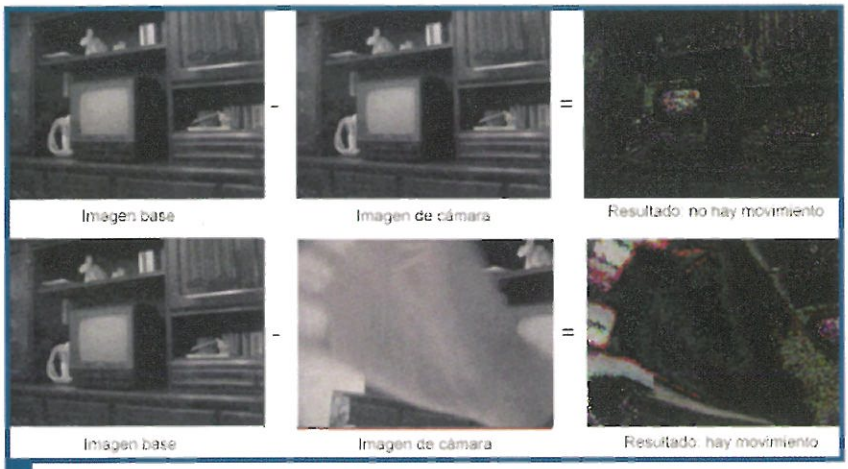


Figura 3.- La imagen muestra el resultado de comparar dos imágenes con una misma imagen base.

Si el resultado es mayoritariamente una imagen de color negro, quiere decir que las dos imágenes son prácticamente iguales. Si el resultado contiene un porcentaje elevado (definible por el usuario) de puntos de otro color, se puede considerar que hay movimiento. En la Figura 3, podemos apreciar cuál sería el resultado de comparar dos imágenes con una misma imagen base.

Trabajaremos con imágenes de resolución 192x144 y 15 bits de color

El concepto en sí es fácil de entender, y hasta cierto punto, también es fácil de implementar. Eso sí, hay que estar familiarizado con la estructura *TVIDEOHDR*, a la que le dedicamos bastante espacio en el artículo anterior.

DISEÑANDO LA APLICACIÓN

En la aplicación se ha incluido un componente *TJLCVideoPanelNew*, cuatro *TImages*, algunos botones y cuadros de edición. Existe además una etiqueta, situada encima del *TEdit* denominada *Probabilidad de movimiento*. No aparece en la imagen puesto que su propiedad *Caption* contiene la cadena vacía.

Los botones **Abre**, **Cierra**, **Fuente** y **Formato**, permiten conectar el componente (además de iniciar variables, activar/desactivar controles, etc.), desactivarlo, seleccionar la fuente de entrada y el formato de las imágenes, respectivamente. En este caso, se ha optado por trabajar con una resolución de *192x144*, con 15 *bits* de color. Esta resolución es la mínima que da la tarjeta utilizada durante las pruebas. Otros modelos de tarjetas pueden dar otros valores.

Los componentes *TImage* tienen todos el mismo tamaño, es decir, **192x144**, con la propiedad *Autosize* en **False**, pero la propiedad *Strecht* está en **True**. Así, aunque el formato no sea el correcto, las imágenes no "invadirán" el resto de la pantalla. El botón **Iniciar** hace que la aplicación pase al modo *preview*, con lo que daría comienzo el proceso de detección de movimiento.

Los controles **Parámetro**, **Tolerancia** y **Movimiento**, permiten cambiar el comportamiento de la función *BitBlt*, definir la tolerancia (es decir, qué se considera negro o no), y cambiar el valor a partir del cual se considera que hay movimiento (ejemplo: 0,15 indica que si la **probabilidad de movimiento** es mayor o igual a dicho valor hay movimiento).

Los controles **Comparar** y **Comparar Con** están muy relacionados. El segundo de ellos se usa para establecer si la diferencia de imágenes se realiza entre la imagen base y la actual, o entre una imagen capturada anteriormente y la actual. Si se elige comparar con la anterior, la imagen actual se compara con la *i-esima* anterior (donde **i** viene dada por el control **Comparar**).

Los únicos controles que tienen código asociado a alguno de sus eventos son **Probabilidad de movimiento** y el componente de vídeo. El primero de ellos cambiará el texto de la etiqueta usada para avisar que existe movimiento, cada vez que su contenido cambie, si el valor de éste es mayor que el almacenado en el control **Movimiento**. El componente de vídeo tiene código asociado a dos de sus eventos: *OnYield* (un simple *Application.ProcessMessages*) y *OnFrame*.

EL EVENTO ONFRAME

Este código se activará, desde modo *preview*, una vez por cada *frame* que se reciba (en este caso se ha optado por trabajar a 10 *f.p.s*). Constituye el centro de la aplicación. Se puede observar en el Listado 3, también incluido en el *CD-ROM*.

El código asociado al evento OnFrame inicia el proceso de comparación de imágenes

La variable auxiliar **CadaCuantas** se utiliza para saber cuándo hay que actualizar el control **Imagen anterior.** Esta actualización depende del valor que hayamos introducido

en el control **Comparar.** A continuación se realiza una llamada a la función *SetDIBits*.

Toda la información de la imagen DIB está contenida en la variable BmpHead

Como ya vimos el mes pasado esta función es muy importante, ya que toma los datos en formato *DIB* (que obtiene de la estructura *PVIDEOHDR*, puntero a *TVIDEOHDR*), y los copia al *bitmap* incluido dentro del *TImage* **Imagen Actual**, teniendo en cuenta el tamaño, profundidad de color, etc. que se le indica mediante la variable **BmpHead.**

Por ultimo, dependiendo de si la comparación es con la imagen base o con la anterior, llamaremos a la función **diferencia** con unos parámetros u otros.

LA FUNCIÓN DIFERENCIA

El código de esta función es el que se muestra en el Listado 4, como todos los demás incluido en el *CD*.

Como se puede observar no está completa. Dentro del *case* hay unas 15 posibilidades, todas idénticas. La única diferencia reside en el parámetro que indica a la función *BitBlt* la operación que debe realizar con las imágenes. La que se muestra en el listado, concretamente, se encarga de restar dos imágenes que es, precisamente, lo que necesitamos para nuestros propósitos. Es posible elegir la operación a realizar mediante el *combo* **Parámetro BitBlt.**

La función *BitBlt* realiza una transferencia de bloques (y si hay suerte, y nuestra tarjeta de vídeo lo soporta, acelerada por *hardware*), será bastante rápida). Los parámetros que recibe son: *device context* de la imagen destino, coordenadas **x** e **y** de la esquina superior izquierda del rectángulo destino, ancho y alto de los rectángulos fuente y destino, *device context* de la imagen fuente, coordenadas **x** e **y** de la esquina superior izquierda del rectángulo fuente, y por último, la operación que se va a realizar.

La función BitBlt se encarga de restar las dos imágenes

Como son muchas, remito al lector a la ayuda de *Delphi*. En pocas palabras, lo que hace esta función es: **Destino = Destino operación Fuente.** Por último, se actualiza el control **Probabilidad de movimiento** (cuyo valor es igual a **1** menos el porcentaje de puntos negros de la imagen).

LAS FUNCIONES CALCULANEGROS Y EVALUA

El código se muestra en el Listado 5, incluido en el *CD*. La función *Evalua* recibe dos *bytes*, que une y convierte en un valor de tipo *word*. Dicho valor (16 *bits*) corresponde a un índice de color de la paleta lógica de colores. Este valor lo usaremos para obtener un color físico de 32 *bits* (mediante la función del *API Win32 PaletteIndex*). A partir de éste obtendremos los valores de los componentes rojo, verde y azul. Si los tres componentes están por debajo de la tolerancia (variable **TOLV**, donde **TOLV = Tolerancia.Value**), se considera a dicho valor como "negro".

NOTA: No intente usar Tolerancia.Value en vez de TOLV. La aplicación se volvería muy lenta, hasta el punto de que no sería capaz de funcionar a 10 f.p.s. (o por lo menos en un PII 266 no ha sido capaz).

Esta función se ejecuta exactamente 27648 veces (192*144) por cada *frame*, por lo que conviene que sea lo más rápida posible, y el acceso a la propiedad *value* no es todo lo rápido que se podría esperar.

La función **CalculaNegros** recorre toda la imagen (mediante un bucle *while*) línea a línea, evaluando cada uno de los puntos. Al final devolverá el porcentaje de puntos negros que hay en la imagen con respecto al total.

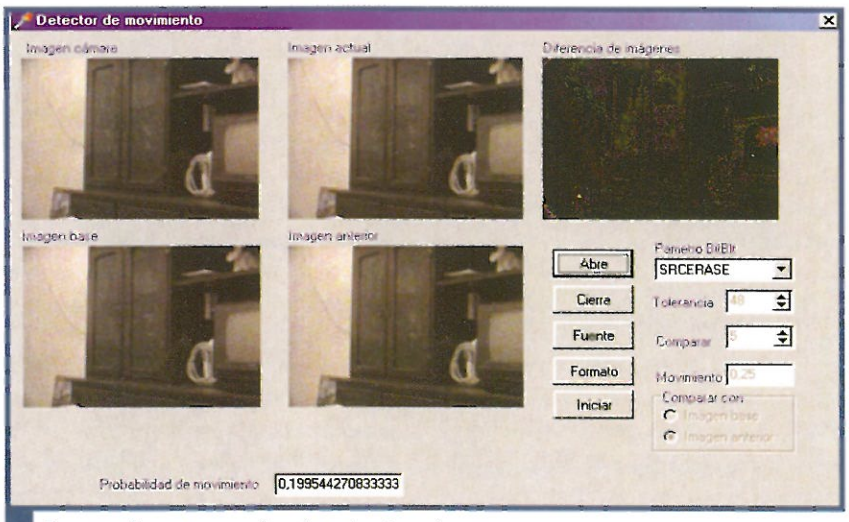


Figura 4.- En esta imagen la aplicación aún no detecta movimiento.

El tamaño del buffer es *BitMap.Width*2*, debido a que cada punto de la imagen se almacena como un *word* (cada punto ocupa dos *bytes*, puesto que usamos 15 *bits* de profundidad de color para las imágenes de vídeo).

La función CalculaNegros indica el porcentaje de puntos negros en la imagen

Otra cosa más: estas funciones no están preparadas para trabajar con más colores. No fallan, pero tampoco producen resultados correctos. Para nuestras pruebas, 15 *bits* de profundidad de color son más que suficientes.

FUNCIONAMIENTO

Con lo explicado anteriormente, el lector ya se habrá hecho una idea de cómo funciona la aplicación. Veamos ahora los resultados.

Ejecute la aplicación, y pulse el botón **Abre.** Deben rellenarse todos los *TImage* a la vez. Si no lo hacen (todos presentan una imagen azul), elija la fuente de entrada adecuada (*TV*, *composite*, etc.), pulse **Cierra**, y después de nuevo el botón **Abre**. A continuación seleccione el formato. Recuerde: 192x144 (o similar), y 15 *bits* de color. Pulse el botón **Iniciar**. En la Figura 4 puede ver una imagen de la aplicación en funcionamiento. Por ahora no se detecta movimiento. Pruebe a pasar la mano por delante de la cámara, o provoque movimiento de cualquier forma. El resultado de estas acciones se puede apreciar en la Figura 5.

Pruebe a cambiar la tolerancia de color, el valor a partir del cual se considera movimiento, etc. Sin embargo unos valores serán más apropiados que otros, dependiendo de la situación de la cámara, condiciones de luz, etc.

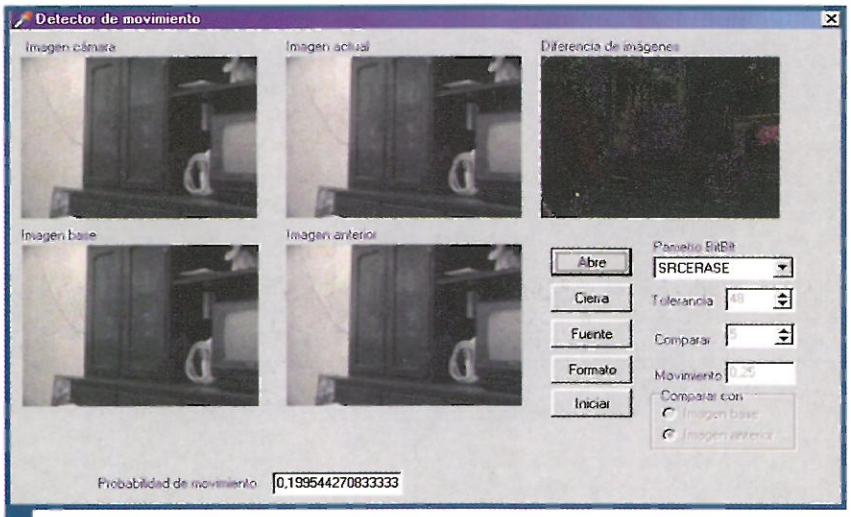


Figura 5.- Ahora la aplicación sí ha detectado movimiento.

WEB'S DE INTERES

En las siguientes direcciones *web* es posible encontrar más información relacionada con la captura y visualización de vídeo:

www.kolban.com
msdn.microsoft.com
www.program.com
www.connectix.com

Grupos de news:
comp.os.ms-windows.programmer.win32
comp.os.ms-windows.programmer.multimedia
es.comp.lenguajes.delphi

En www.clubdelphi.com y jlceada.virtualave.com podréis encontrar el componente, así como futuras actualizaciones. Además, en el *CD* que acompaña a la revista, se incluye una nueva versión que añade nuevas propiedades, destinadas a cambiar el formato del audio capturado.

CONCLUSIÓN

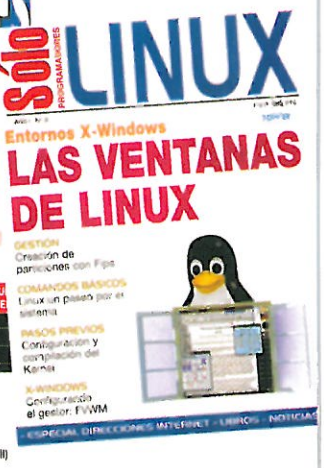
En esta entrega hemos demostrado que, con poco trabajo, es posible sacar partido al componente que hemos desarrollado. Evidentemente, nadie se va a hacer millonario con las dos aplicaciones que se han creado, pero creo que son suficientes para entender su funcionamiento, y para saber cómo trabajar con los datos que capturamos.

Además, considero que podría ser un buen punto de partida para futuros proyectos. Por ejemplo, podría ser interesante mejorar la aplicación de vídeoconferencia, de forma que se ajustase a los estándares existentes en el mercado para la transmisión de audio/vídeo. O mejorar el algoritmo de detección de movimiento, para que nuestra aplicación de prueba funcione en ambientes más "hostiles". Aquí ya entran en juego la imaginación y las ganas de trabajar y curiosear que tenga cada uno.

¿TE FALTA ALGÚN NÚMERO DE...?

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES



Nº 51: AGOTADO

BOLETÍN DE PEDIDOS

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.
(Revista Sólo Programadores). C/ San Sotero, 5. 1ª Planta. 28037 Madrid
Tlf: 91 304 87 64. Fax: 91 327 13 03

Deseo que me manden los número/s.....
.....
.....

NOMBRE Y APELLIDOS:.....
.....

EDAD:..... PROFESIÓN: .....TFNO: .....
DOMICILIO: .....
CIUDAD: .....
PROVINCIA: .....C.P: .....

❑ **Precio por unidad: 975 pesetas** + 700 ptas. de gastos de envío.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Talón bancario a nombre de REVISTAS PROFESIONALES. S.L
❑ Domiciliación bancaria
❑ Contra reembolso
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41. Nº c/c: 0075/1040/43/0600047439

DATOS DE DOMICILIACIÓN:

Banco..... FIRMA

Domicilio.....

Nº de Cuenta.....|.....|.....|.....

Titular.....

Fecha.....

DNS a fondo (I)

Vicente Antonio Sánchez Werner (thewind@maptel.es)

DNS es uno de los más importantes protocolos de Internet, pero un gran desconocido para la mayoría del público. En esta nueva serie lo analizaremos con profundidad desde los puntos de vista del usuario y del servidor.

INTRODUCCIÓN

En *Internet* se trabaja sobre una serie de ficciones que facilitan nuestro uso y comprensión. La ficción básica es la identificación de ordenadores mediante nombres, que en realidad son arbitrarios y deben ser relacionados de alguna forma con la identificación real del ordenador en la red.

En la pila de protocolos (abreviadamente *TCP/IP*) que forman la base de *Internet* no se utilizan nombres, sino una identificación numérica única (dirección *IP*, formada por una serie de números del tipo xxx.xxx.xxx.xxx) que resulta difícil de recordar, pero muy sencilla para los ordenadores. Cada uno de estos números puede establecerse en el rango 0 a 255. Debido a la dificultad asociada a las direcciones IP, se creó el servicio de asignación de nombres (*DNS*: *Domain Name System:* Sistema de Nombres de Dominio), para que fuese mucho más natural para los usuarios el recordar a qué ordenador se querían conectar.

ORIGEN Y EVOLUCIÓN DEL DNS

El nacimiento del *DNS* es fácilmente comprensible si consideramos el origen y desarrollo de *Internet* hasta los años 80.

En 1969 se creó el antecedente directo de *Internet*, que no usaba *TCP/IP* sino un protocolo anterior; en aquel momento existían muy pocos ordenadores con conexión a *Internet*. De los dos ordenadores que compusieron la red inicialmente se paso a 4 a finales de 1969, y ese lento crecimiento siguió durante la década de los 70.

Durante aquella época, se utilizaba un fichero llamado *hosts* para almacenar la correspondencia entre las direcciones lógicas (que eran las que se utilizaban en el protocolo de red y luego en *TCP/IP*) y las físicas o reales, que eran implementadas en los dispositivos de red.

DNS es el protocolo que transforma los nombres de los ordenadores en direcciones IP

En 1973, se ideó la primera versión del protocolo *TCP/IP* contando con que, en el futuro, sólo habría redes de carácter nacional, mientras empresas como *Xerox* desarrollaban en sus laboratorios lo que hoy son las *LAN* (redes locales).

A finales de los 70, aparecieron las *LAN* y el número de redes y ordenadores conectados a *Internet*

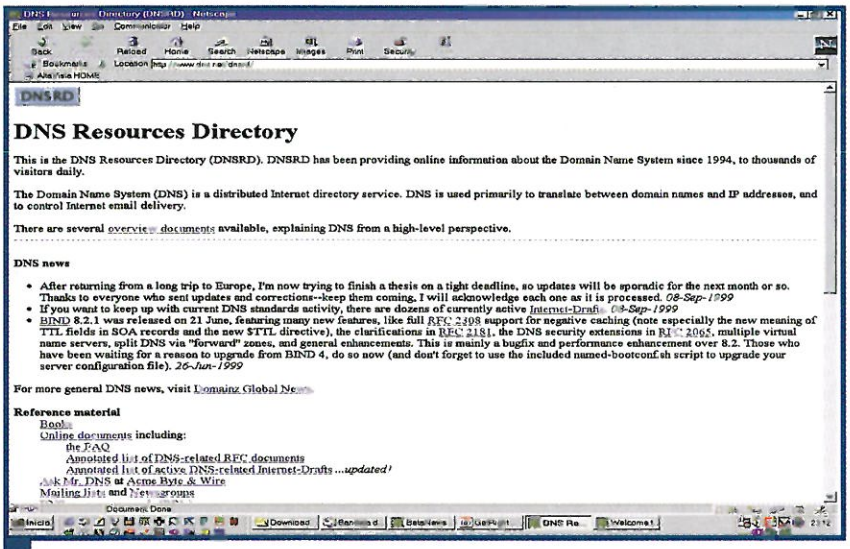


Figura 1. Una página con información referente a DNS

creció de forma desorbitada. En ese punto se hizo patente la necesidad de asignar nombres a los ordenadores, para facilitar el acceso y el uso de los mismos a las personas. La respuesta a esta necesidad llegó en los principios de los años 80, a través del servicio de *DNS*.

El antiguo archivo *hosts* que contenía todas las correspondencias entre nombres y direcciones *IP* se vio rápidamente superado por el imparable crecimiento de los ordenadores conectados a *Internet*. Pensemos que cada vez que se conectaba uno nuevo, había que añadir sus datos a los archivos de todos los ya existentes, así como colocar una copia del archivo *hosts* en el recién incorporado. Este sistema resultaba poco eficiente además de ser lento y muy costoso.

El siguiente paso fue colocar este archivo en un ordenador que actuaba como servidor y que resolvía los nombres (daba la dirección *IP* para un nombre dado) para todos los ordenadores conectados. Este esquema rápidamente se vio superado y finalmente se adoptó el actual sistema de *DNS* durante el final de 1983 y el comienzo de 1984 con espacio de dominios jerarquizado.

Tradicionalmente, *UNIX* ha sido la plataforma preferida para la implantación de este servicio, existiendo gran cantidad de *software* para este sistema operativo y por extensión, para Linux. Hoy en día, *Linux* es usado con frecuencia en servidores *DNS* debido a sus características de estabilidad, fiabilidad y reducido coste económico.

CONCEPTOS Y DEFINICIONES BÁSICAS

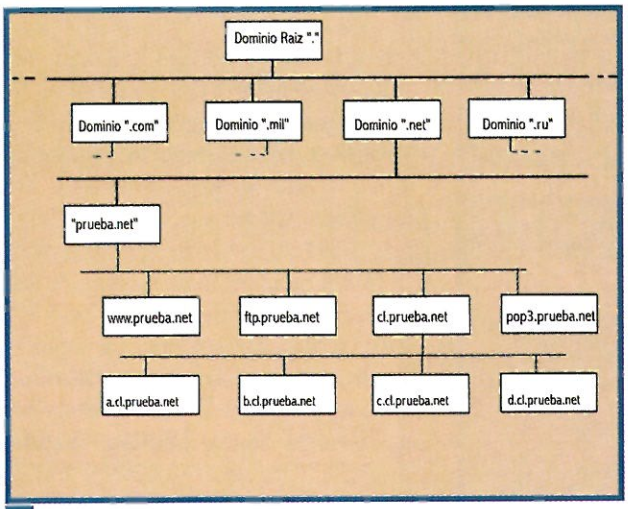
Antes de adentrarnos en la utilización del *DNS*, hemos de tener en cuenta su importancia y complejidad, así como conocer los diversos conceptos que se utilizan. La *IETF* (Organización que define los protocolos en uso en *Internet*) publica la documentación completa del *DNS* en unos documentos denominados *RFC* en los que se comentan todos y cada uno de los aspectos del protocolo. Para hacernos una idea de su complejidad baste decir que 77 *RFC* están dedicados exclusivamente a documentar el protocolo *DNS* y las ampliaciones que ha sufrido a lo largo del tiempo.

El concepto más simple es el concepto de *nombre de un ordenador* (*host name*), formado por un conjunto de caracteres al que le corresponde una dirección *IP*. Esta cadena de caracteres ha de cumplir ciertas normas (detalladas en el *RFC* 1137) entre las que se incluye la imposibilidad de contener signos de puntuación (ejemplos válidos serían: **mega**, **galadriel**, **3ti34com**, **alpha5**; mientras que inválidos serían **i23**, **a!oo**, **a$sign** y similares). Existen otras normas que se aplican, pero no con rigurosidad en la práctica, aunque si de-seamos implementar un servicio de *DNS* privado, independiente de *Internet* podremos obviar estas limitaciones.

Este protocolo es de los mas complicados, su documentación ocupa cerca de 2.000 páginas

El siguiente concepto que conviene aclarar es el de *dominio*. Se podría definir como un nombre que agrupa ordenadores que poseen algún tipo de relación entre sí; por ejemplo, la pertenencia física a una universidad, la localización geográfica, o la orientación comercial. Los dominios incluyen nombres concretos de ordenadores, y se separan de ellos mediante un '.', por ejemplo: **galadriel.soloprogramadores** donde indicaríamos que el ordenador **galadriel** pertenece al dominio **soloprogramadores**.

A su vez los dominios pueden estar anidados; es decir, un dominio



Esquema 1. Jerarquía de nombres y dominios.

también puede contener en sí mismo a otros dominios (llamados de segundo o tercer nivel, según el nivel de inclusión), siempre concretando de derecha a izquierda los de mayor nivel, por ejemplo: **galadriel.soloprogramadores.es**; la cadena de *host name* y los dominios en los que está incluido se conocen como nombre completo cualificado (*FQDN*). Esta característica permite establecer una jerarquía de nombres y dominios (ver *Esquema 1*) que se refleja en la estructura de los servidores y sus conexiones mutuas, además de optimizar el acceso y el trafico de red requerido por las consultas de *DNS*.

En resumen, *DNS* es un sistema jerárquico, y como toda jerarquía ha de tener una cima. La cima del espacio de dominios es el dominio raíz que se simboliza por "." Este dominio raíz es la cumbre de la que cuelgan todos los dominios formando el espacio de dominios. Inmediatamente bajo este dominio, encontramos los *TLD's* o nombres de dominios globales. Pueden ser de tres tipos:

- *Dominios organizativos*. Utilizan un código de tres letras que indica la actividad primaria que desempeña la organización dueña del mismo. Actualmente sólo hay autorizados 7 de estos códigos, algunos de los cuales están restringidos para el uso único y exclusivo de organizaciones y entidades norteamericanas. La escasez de estos dominios y las restricciones que operan sobre ellos han hecho que en repetidas ocasiones se haya propuesto la incorporación de más *TLD's*, aunque todavía no se han registrado. Los *TLD's* actuales pueden ser verse en la Tabla 1 y los propuestos en la Tabla 2.

La cantidad de nombres reconocibles por el público es limitada y, hoy por hoy, el número de TLD's es insuficiente

- *Dominios geográficos*. Indican la localización geográfica y son utilizados por todos los países, excepto EEUU, que suele acaparar los *TLD's* organizativos. El servicio que controla la asignación de los dominios en ese país puede subdividir el espacio de dominios ofreciendo dominios de segundo nivel (del tipo: **co.uk**, **ac.uk**, **ac.at**). La lista completa de los dominios geográficos de primer nivel se encuentra en la norma *ISO-3166*, aunque algunos de ellos no están implantados o en uso. La escasez de dominios *TLD's* organizativos ha llevado a que algunos países del Tercer Mundo vendan sus dominios a compañías que los explotan comercialmente a cambio de unas mínimas estructuras como pueden ser *Tuvalu* (.tv) o *Tonga* (.to).

- *Dominios inversos*. Estos dominios no suelen ser conocidos por el público en general, ya que se usan sólo en condiciones muy concretas y generalmente de forma completamente oculta para el usuario. Su función es la de permitir la resolución inversa en el *DNS*; es decir, partiendo de una *IP* obtener el nombre completo. El nombre de este dominio es: *.IN-ADDR.ARPA.*

Existen una serie de entidades que se encargan de gestionar el espacio de nombres, algunas de las cuales no tienen ánimo de lucro. Actualmente se está efectuando un cambio en la estructura de las mismas, y la *IANA* que antes controlaba todos los *TLD's* (a través de *Internic*, que cedía la explotación exclusiva de los dominios comerciales a una empresa americana llamada *Nestwork Solutions*) y la asignación de direcciones de *Red*, ha cedido su puesto al frente del *DNS* a *ICANN*. Este cambio es importante ya que EEUU cede parte de su control a una organización sin ánimo de lucro y con presencia internacional, con lo que se liberaliza el mercado de los registros de los *TLD's*.

Independientemente de estas asociaciones que operan en el ámbito internacional, existen organismos y empresas a un nivel nacional, ofreciendo los servicios de registro para el dominio nacional y facilitando los tramites de registro de los *TLD's*, siempre contando con la autorización del *ICANN*.

Al existir escasez de *FQDN's* representativos posibles (nada impide registrar **www.xdfghjhg.com**,

Tabla 1. TLD's reconocidos por ICANN y su significado

.gov	Reservado a organizaciones y/o entidades dependientes del gobierno de EEUU.
.mil	Organizaciones y redes pertenecientes a los cuerpos y organizaciones militares americanas
.edu	Organizaciones de orden educativo, únicamente usado por entidades de EEUU.
.net	Utilizado por entidades, organizaciones y compañías de redes. Actualmente también se comercializa y puede ser registrado por personas particulares.
.com	Compañías y entidades privadas con animo de lucro, personas particulares y otras entidades.
.int	Organizaciones internacionales creadas por tratados internacionales.
.org	Organizaciones sin animo de lucro, también utilizado por personas particulares.

Tabla-2. TLDs propuestos para la aprobación por ICANN y significado.

firm	Compañías comerciales exclusivamente.
.store	Compañías comerciales con tiendas online y tiendas online exclusivamente.
.web	Entidades y organizaciones relacionadas con la promoción y el desarrollo de la WWW.
.rec	Entidades y compañías relacionadas con el mundo del espectáculo y el ocio en general.
.arts	Entidades, compañías y personas relacionadas con el mundo de las artes.
.nom	Exclusivamente para denotar dominios de personas individuales
.info	Empresas y entidades que ofrezcan diversos sistemas de información o informaciones en general.

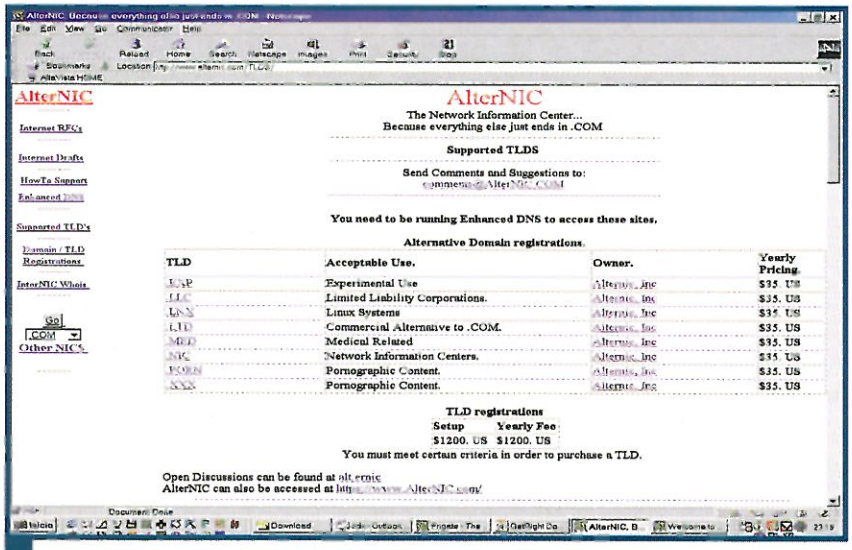


Figura 2. Pagina de alternic. Uno de los servicios de DNS alternativos

pero su representatividad y la dificultad de recordarlo la hacen inoperante), no han tardado en aparecer empresas que ofrecen registros de *TLD's* no reconocidos por *ICANN* y que amplían el número de posibilidades de registro. El inconveniente es que aquellas personas que deseen acceder a máquinas que usen estos servicios deberán modificar manualmente sus *DNS* para que apunten a los de las máquinas que ofrecen el servicio; o, en su defecto, acceder a las máquinas con el antiguo recurso de la dirección *IP* numérica.

Algunas de estas empresas han realizado polémicas acciones de protesta (*Alternic* hizo que **www.internic.org** apuntase a su pagina *Web* como protesta por la política de ésta) y ofertas de registro sin precedentes (*TLGDM* regaló las cuotas de registro a quienes se registrasen con ellos el día 23 de agosto) que hasta el momento apenas han tenido efectos. En la Tabla 3 podemos encontrar información sobre algunos de estos servicios.

Continuando con las definiciones básicas tenemos que un *servidor DNS* es aquel ordenador que posee el control de una zona o dominio, y que es capaz de resolver los nombres de dicho dominio. Puede estar clasificado en diversas categorías, según sean sus funciones y utilidad.

Un *servidor DNS* es *primario* de un dominio si y sólo si, contiene los datos de ese dominio determinado y tiene la potestad de modificarlos, además de ser capaz de efectuar un servicio normal de resolución de nombres. Un *servidor secundario* posee las mismas características, excepto la potestad de modificar los datos de un dominio. Esta distinción entre primario y secundario aparece por motivos de redundancia y de tolerancia al fallo, ya que si uno de los servidores falla, el otro puede

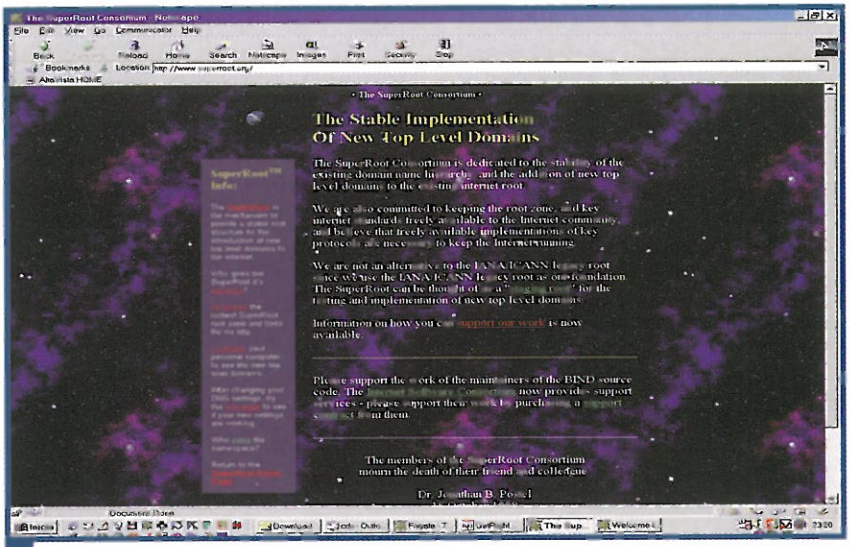


Figura 3. Pagina de Superroot, organización de presion de nuevos TLDs

Tabla-3. Organizaciones alternativas de registro de TLDs. www.alternic.org

Alternic.	Registra dominios alternativos a los TLD's de ICANN
www.eu.org Eu.org.	Registra dominios del tipo jamaica.de.eu.org de forma gratuita. Muy activa en la demanda de nuevos TLD's.
www.superroot.org	Organización de registro alternativa. Registra dominios alternativos.
www.open-rsc.org	Organización de presión que promueve la creación de nuevos TLD's.
www.tldns.com	Compañía que registra nuevos dominios alternativos.

seguir funcionando y dando el servicio sin que el usuario se dé cuenta del problema.

Un servidor que tiene una autoridad reconocida para un dominio, se dice que es *delegado*, ya que los servidores del nivel superior a él le ceden parte de su autoridad y responsabilidad en el manejo y mantenimiento del espacio de dominios con respecto al dominio particular.

Un servidor que es delegado, primario y da un correcto servicio de *DNS* se dice que es un *servidor autorizado*. Es posible hacer un servidor autorizado no delegado, pero eso restringiría el uso de sus servicios a aquéllos que lo tienen puesto explícitamente en sus listas de consultas de *DNS*.

Las consultas que podemos efectuar a un servidor *DNS* o que un servidor puede hacer a otro son de tres tipos diferentes y de ellas se obtienen resultados diferentes.

- *Consultas recursivas*. Este es el tipo más frecuente de las realizadas por el cliente. El objetivo de la misma es darnos la *IP* del nombre exacto que nosotros proporcionamos, y por lo tanto nos dará la *IP* o nos dará un error si no encuentra una correspondencia exacta.

- *Consultas iterativas*. Suele efectuarse entre servidores y el objetivo de la misma es ofrecernos siempre una dirección *IP* que, en caso de no ser la correcta, se corresponderá con la de un servidor de *DNS* asociado al dominio del *FLQDn* que hemos solicitado.

Existen una serie de empresas que efectúan registros de TLD's no reconocidos

- *Consultas inversas*. Pretenden conseguir lo opuesto a las anteriores, es decir, conseguir un *FLQDn* partiendo de una dirección *IP*. Sin embargo, tal y como se implementó el *DNS*, no es posible hacer esta operación sin efectuar una búsqueda por todos los servidores *DNS*; es obvio que una búsqueda de este calibre gastaría una cantidad ingente de recursos de *Red* y tiempo, por ello se creó el dominio *IN-ADDR.ARPA*. Incluye en él todas las direcciones *IP* posibles y es el que permite que una resolución inversa se resuelva igual que las normales. Todas las máquinas con un nombre asignado, tienen una denominación que pertenece a este dominio; este nombre se anota escribiendo al revés la dirección *IP* de la máquina y añadiendo al final el dominio inverso. Por ejemplo imaginemos el siguiente ordenador:

```
FLQDn: wild.soloprogramadores.es
IP:    195.54.98.12
```

Su *FLQDn* inverso es:

```
12.98.54.195.IN-ADDR.ARPA
```

El dominio inverso es:

```
98.54.54.195.IN-ADDR.ARPA
```

Este último punto sería cierto si el dominio **soloprogramadores.es** estuviese contenido exclusivamente en la *Red* **195.54.98.xxx.**

Como podemos ver, si cada máquina tiene un nombre de dominio de este tipo, al efectuar una resolución inversa, el funcionamiento será idéntico al de una resolución normal, pero obtendremos el *FLQDn* y no una *IP*.

El dominio .IN-ADDR.ARPA permite y facilita las resoluciones inversas

En la actualidad aún quedan restos del pasado pre-*DNS* en funcionamiento, como es el archivo *hosts* o la noción de *resolver*. Un *resolver* es un programa cliente que se encarga de transformar nombres en direcciones numéricas y viceversa. En la actualidad no hay apenas *resolvers* de uso generalizado; la mayoría están incorporados como funciones de librería en pilas de *TCP/IP* o en librerías de programación, quedando relegados a ser herramientas de diagnóstico o pequeñas utilidades poco conocidas por el público en general.

El otro resquicio del pasado que aún queda en pie, el archivo *hosts*, es un archivo que contiene información para resolver nombres de ordenadores en una red; siendo un pequeño y poco usado rescoldo del antiguo sistema de resolución de nombres.

El espacio de los dominios es jerárquico y esto se refleja en la estructura de la *Red* y en su funcionamiento (que veremos más adelante); baste saber por el momento que cada servidor *DNS* se corresponde con un nodo de una estructura en forma de árbol, en cuya cima se encuentran los servidores raíz o *root* que gestionan los dominios de más alto nivel.

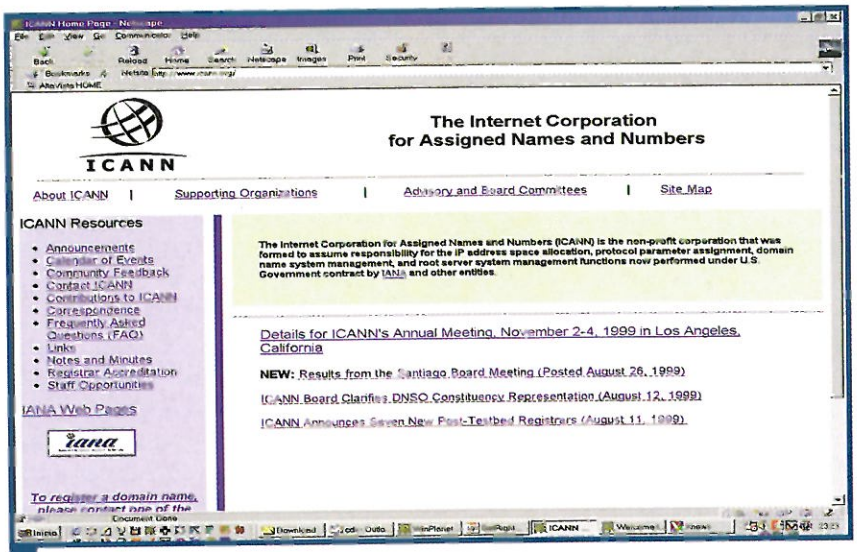


Figura 4. Pagina de ICANN

FUNCIONAMIENTO

Hasta ahora habíamos visto los conceptos más básicos que eran asequibles sin un conocimiento del funcionamiento exacto del *DNS*. Una vez más hemos de remontarnos a los inicios de *Internet* para poder comprender el funcionamiento del sistema.

El fichero *hosts* guarda, con una sintaxis muy simple, la misma información que el *DNS*; es decir, guarda la información que permite relacionar un nombre de un ordenador con una dirección lógica de *Red*. Este archivo permite al ordenador resolver por sí mismo nombres, pero como es obvio, es ineficaz y pesado incluir en él todos los ordenadores de *Internet*, pero tiene su utilidad como un pequeño caché; además de ser una buena solución para reducidas redes en las que el numero de ordenadores es bajo y la movilidad de los equipos a lo largo de la *Red* pequeña. Un ejemplo de la sintaxis de este archivo puede verse en el Listado 1.

El archivo hosts es un resto del pasado, pero puede ser de utilidad en pequeñas redes

Cuando efectuamos una consulta sobre un nombre y utilizamos el fichero *hosts* como única fuente de nombres, el servidor de *DNS* o el *resolver* se limitan a escanear el archivo de arriba abajo hasta encontrar una correlación entre el nombre y una *IP*. En caso de no existir, nos daría un fallo indicando que no ha sido posible encontrar el ordenador.

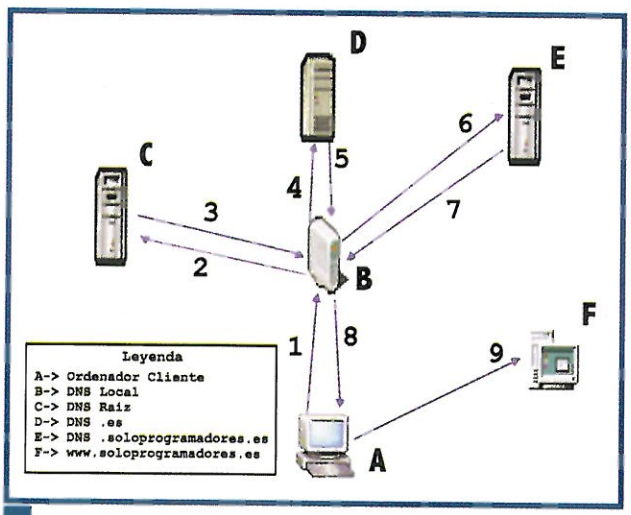
El funcionamiento del *DNS* es más complicado y más lento, ya que en numerosas ocasiones la resolu-

ción de un nombre requiere la consulta a más de un servidor, además de que el funcionamiento es diferente según le hagamos una petición u otra. La mejor forma de ver el funcionamiento es con una búsqueda ejemplo en *Internet* y ver cómo se realizaría la misma.

Incluso si no hay problemas la operación de búsqueda requerirá 9 pasos

Como podemos ver en el Esquema 2, cuando nuestro ordenador envía una petición al servidor de nombres local pidiendo para resolver el nombre de **www.demo.soloprogramadores.es** se producen los pasos siguientes:

1) El cliente hace una consulta recursiva a su *DNS* sobre **www.demo.soloprogramadores.es**.
2) El servidor de *DNS*, a su vez, efectúa una consulta iterativa a uno de los servidores *DNS* raíz solicitando un servidor autoritario para el primer dominio, el genérico "**.es**".
3) El servidor raíz da a ese cliente una respuesta indicándole un servidor autorizado para el dominio **.es**.
4) Con esta respuesta el servidor local vuelve a enviar una consulta iterativa al servidor **.es** para conseguir una lista de servidores autorizados para el dominio **soloprogramadores.es**.
5) El servidor contesta con la lista de servidores *DNS* para el dominio **soloprogramadores.es**.
6) EL *DNS* local se conecta a uno de los servidores *DNS* autorizados del dominio y efectúa una búsqueda iterativa para la dirección de **www.demo.soloprogramadores.es**.
7) El *DNS* de **soloprogramadores.es** da la dirección *IP* de la máquina **www.demo.soloprogramadores.es** ya que en este caso, dicha máquina está en su información de zona y dominio.
8) El *DNS* local da a la máquina que ha realizado la consulta inicial la dirección *IP* de la máquina.
9) Con esta dirección el cliente pasa a conectarse a la máquina correspondiente para efectuar la operación que en un primer momento se había propuesto hacer la máquina cliente.



Esquema 2. Envío de petición al servidor.

Como podemos observar, la operación completa de búsqueda ha llevado nueve pasos básicos, entre los que no hemos tenido en cuenta la posibilidad de que alguno de los servidores *DNS* no funcionase; en el caso de que en uno de los servidores (digamos el *DNS* para **.es**) no funcionase, el *DNS* solicitante haría una búsqueda iterativa de nuevo y cogería el nombre siguiente al que hubiera fallado: he aquí el porqué de la existencia de al menos dos *DNS* por cada dominio, de esta forma el cliente podría llegar a la máquina deseada aunque el *DNS* primario o el secundario estuviesen fuera de servicio.

El funcionamiento de la consulta inversa, es exactamente igual, ya que el cliente transforma automáticamente la *IP* en el dominio que ha de ser buscado y la serie de búsquedas y respuestas es idéntica a la de una búsqueda normal.

Existen tres tipos de consultas al DNS, cada una nos da un resultado diferente

Este es el funcionamiento básico del *DNS* a un nivel operacional, visto desde el cliente. Sin embargo los servidores *DNS* permiten efectuar operaciones más complejas y posibilidades que comentaremos cuando hablemos de los servidores y de su implementación.

RESUMEN

En este artículo hemos visto los conceptos básicos del *DNS*, así como su funcionamiento operativo. En los próximos artículos de la serie veremos qué utilidades existen en *Linux* para buscar información en el *DNS*, o cómo, partiendo de una *IP*, podemos llegar a obtener un nombre; también veremos cómo implementar un servidor *DNS* en *Linux*, así como implementar prestaciones más avanzadas (*DNS* dinámico, Cache, etc.).

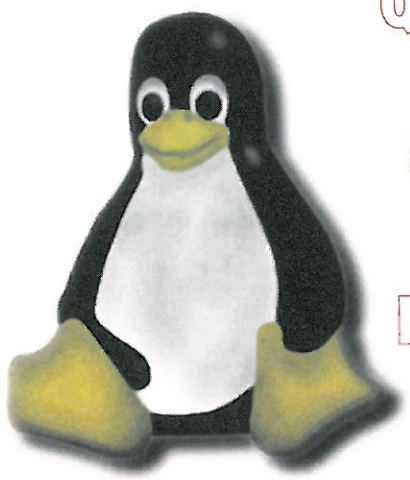
SUSCRIPCIÓN DOBLE

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

SÓLO PROGRAMADORES LINUX

QUE NO TE LÍEN CON EL <CÓDIGO>



PARA NO QUEDARSE HELADO CUANDO HABLEN DE LINUX

BOLETÍN DE SUSCRIPCIÓN

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.
C/ San Sotero, 5. 1ª Planta. 28037 Madrid. Tlf: 91 304 87 64. Fax: 91 327 13 03

Quiero suscribirme a la revistas SÓLO PROGRAMADORES y SÓLO PROGRAMADORES LINUX desde los Nºs .....y beneficiarme de las condiciones de estas magníficas promociones:

❑ **SUSCRIPCIÓN ANUAL**
24 NÚMEROS + 24 CD-ROMs
AL PRECIO DE
13.720 ptas. / 82,45 €

❑ **SUSCRIPCIÓN ANUAL ESPECIAL ESTUDIANTES**
24 NÚMEROS + 24 CD-ROMs
por sólo 10.876 ptas. / 65,36 €

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❑ Domiciliación bancaria
❑ Contra reembolso

Soy antiguo suscriptor
❑ **Sí** ❑ **No**

PARA ENVÍOS AL EXTRANJERO SÓLO SE ADMITIRÁN LAS SIGUIENTES FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❑ Eurocheque conformado con un banco español a nombre de REVISTAS PROFESIONALES, S.L

NOMBRE Y APELLIDOS:.....

EDAD:..... PROFESIÓN: .....

TFNO: ..... DOMICILIO: .....

CIUDAD:..... C.P: .....PROVINCIA:.....

Promoción válida hasta agotar existencias

Datos de domiciliación:

Banco.....

Domicilio .....

Nº de Cuenta.....

Titular.....

Fecha .....

FIRMA

Windows NT 4.0 e IIS (1)

Jordi Agost Moré (agost@eup.udl.es)

Las siglas NT suenan cada vez más en todos los ámbitos ¿Qué es? ¿Qué diferencias tiene con Windows 95/98? ¿Cuántas versiones existen? ¿Cuáles son sus diferencias? Eso es lo que intentaremos descifrar en las próximas páginas, con esta nueva sección que, a partir de ahora, encontraréis de forma habitual.

ORÍGENES DE WINDOWS NT

Aunque parezca curioso *Windows NT* no proviene de una ampliación de ningún *Windows* de 16 *bits*. El primer origen al que debemos remontarnos es al sistema *VMS* de *Digital*, con el que comparte filosofía y parte de la construcción. Esto es debido a que *Microsoft* quería realizar un sistema operativo para los tiempos venideros. Para ello incorporó en su plantilla a *Dave Cutler* (desarrollador de *RSX-11M* y *VMS*) y le encargó la dirección de *NT*.

El nuevo sistema operativo debía estar integrado en un entorno gráfico, ser fiable, seguro, multitarea, escalable, compatible con otros sistemas, multiplataforma y seguir las normas *POSIX*. Todo esto sucedía en 1988, y por aquel entonces el mundo del *PC* aún no estaba muy desarrollado, el lanzamiento se demoró 5 años, momento en el que salió la versión 3.1. Fue un primer paso, pero *Windows NT* aún estaba a mucha distancia de otros sistemas como *Unix*, *Netware* o *AS/400*. En 1994 salió al mercado la versión 3.5 y en 1995 la 3.51, empezando a consolidarse.

¿QUÉ ES WINDOWS NT?

Windows NT (las siglas *NT* se refieren a *New Technology* o nueva tecnología) es un sistema operativo de alto rendimiento basado en los sistemas *Windows*, posicionado por *Microsoft* como el sistema operativo para los negocios empresariales, mientras *Windows 9x* es el sistema operativo para los usuarios domésticos.

Windows NT es un sistema operativo multitarea y multihilo de 32 *bits*. Existen dos versiones de *Windows NT*, *NT WorkStation* y *NT Server*. *NT WorkStation* está diseñado para ejecutar tareas críticas de cliente, mientras que *NT Server* está diseñado para ejecutar tareas de servidor.

Muchas son las características que lo hacen diferente de un sistema operativo como *Windows 9x*, las más importantes son:

- Seguridad.
- Ejecución de aplicaciones *POSIX*.
- Escalabilidad.
- Multiproceso simétrico.
- Soporte para multiplataforma.
- Multitarea y Multihilo.
- Sistema de ficheros

SEGURIDAD

Windows NT también es un sistema operativo seguro, ya que ha alcanzado el nivel *C2* (otorgado por el departamento de defensa y seguridad nacional del gobierno de los *EEUU*) para sistemas operativos, aunque aún así le quedan algunos detalles para alcanzar los niveles más altos. El nivel *C2* presupone:

- Que un usuario posee sus propios recursos (por ejemplo ficheros) y que el usuario establece la seguridad de acceso a los mismos.
- La memoria está protegida. Otros procesos que se ejecutan en la misma máquina no pueden leerla mientras tanto, y tiene que ser liberada inmediatamente después de utilizarla.
- Deben poder establecerse auditorias para las acciones críticas hechas por usuarios en el sistema.
- Los usuarios deben ser capaces de identificarse de manera única hacía el sistema operativo, y esta identificación (con nombre de usuario y *password*) debe permitir el acceso al sistema.

Todo esto significa que *NT* es un sistema seguro, al contrario que *DOS* o *Windows 9x* donde cualquier persona puede acceder a los ficheros y/o recursos del ordenador.

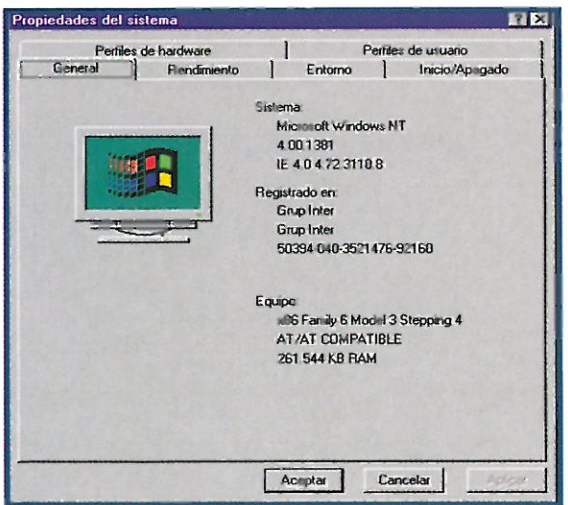


Figura 2.- Pantalla de información general de *Windows NT Server*.

APLICACIONES POSIX

Windows NT soporta aplicaciones *POSIX* (*Portable Operating System Interface o* Interfaz portable del sistema operativo), un conjunto de normas definido por el *IEEE* (*Institute of Electrical and Electronic Engineers* o Instituto de Ingenieros Eléctricos y Electrónicos), para el intercambio de aplicaciones entre distintos sistemas operativos. *Windows NT* soporta el estándar *POSIX 1*, que consiste en una *API* escrita en lenguaje *C*.

El que *Windows NT* posea capacidad multiprocesador significa que en una misma máquina pueden estar instaladas más de una *CPU*.

MULTIPROCESO SIMÉTRICO

Esta característica permite que en una máquina con más de un procesador, éstos puedan trabajar en más de una tarea a la vez. En este tipo de procesador si uno se encuentra sobresaturado puede pasar parte del trabajo a otro.

SISTEMA ESCALABLE

Esto significa que no debemos comprar un nuevo sistema operativo si el rendimiento del nuestro no es satisfactorio. Simplemente debemos instalar más procesadores (*CPU's*), teniendo en cuenta que *Windows NT WorkStation* puede soportar hasta 2 procesadores y la versión *Server* hasta 32.

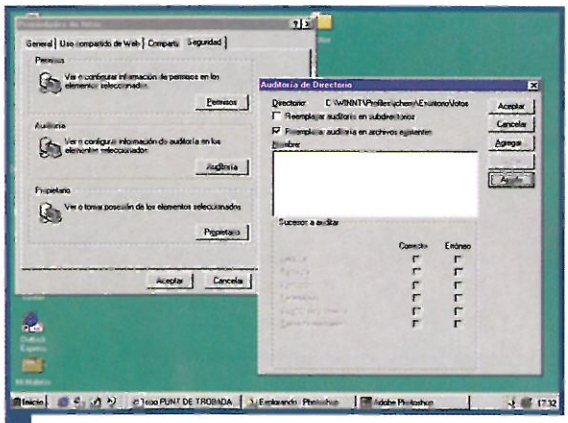


Figura 1.- Opciones de Auditoría de ficheros y directorios de *NT*.

Este hecho es bastante importante para los sistemas basados en una arquitectura cliente/servidor, ya que podemos incrementar de manera muy significativa el rendimiento del sistema aumentando la potencia del servidor en vez de la de cada cliente.

SOPORTE MULTIPLATAFORMA

Windows NT puede ejecutarse en diversas plataformas, desde la plataforma *Intel 80xx* hasta máquinas tipo *RISC* (*Reduced Instruction Set Computers*) como por ejemplo *MIPS R3000, R4000*, así como máquinas *Alpha AXP*.

MULTITAREA Y MULTIHILO

Al contrario que en *Windows 95* y *Windows 3.1* la multitarea es realmente buena. En los sistemas operativos mencionados las aplicaciones son las que dejan el control de la *CPU* a otras aplicaciones. Pero *NT* es el que controla el tiempo, de acuerdo con las preferencias fijadas por el administrador del sistema.

SISTEMA DE FICHEROS Y SISTEMAS OPERATIVOS

El soporte para otros sistemas operativos también está presente. En primer lugar *Windows NT* puede ejecutarse con otro sistema

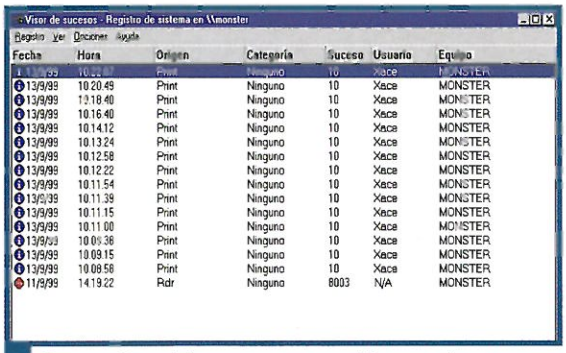


Figura 3.- Control de sucesos por parte del sistema de un usuario.

instalado en la misma máquina, y en segundo lugar puede ejecutar sin problemas la gran mayoría de aplicaciones para *Windows 9x* y *MSDOS*. También puede leer sistemas de ficheros de terceros como por ejemplo *FAT* (el sistema de ficheros de *MSDOS*) o FAT32 (*Windows 95 OSR2* y *Windows 98*). Aunque si utilizamos estos sistemas, perderemos muchas de las características que nos puede ofrecer su propio sistema de ficheros, el *NTFS* (*Windows NT File System*).

NOVEDADES EN LA VERSIÓN 4

En 1996 aparece la versión 4, y con ella los desarrolladores del sistema operativo se encontraron con un gran problema, que surgió al intentar incorporar el sistema gráfico de *Windows 95*. Esto sucedía porque en *Windows NT 3.x* el controlador de vídeo era un dispositivo colocado en un nivel superior al del núcleo, con la fiabilidad que esto supone.

Pero si al incorporar el nuevo sistema gráfico se hacía lo mismo, el sistema se hacía tremendamente lento. Con lo que se decidió incluir el controlador gráfico dentro del núcleo del sistema operativo. Pero esto conlleva el peligro de que si dicho controlador (normalmente desarrollado por terceras firmas) falla, el núcleo del sistema también puede fallar. Dejando aparte dicho problema con la versión 4 se introdujeron también bastantes mejoras, enumeradas brevemente en la Tabla 1.

WORSTATION Vs. SERVER

Windows NT se presenta en dos versiones: *NT Server* y *NT WorkStation*. Brevemente las principales diferencias entre ellos son:

SESIONES ENTRANTES

Una de las principales diferencias es la cantidad de sesiones entrantes que cada sistema operativo permite. *WorkStation* permite hasta 10 conexiones entrantes de *Red*, pero la versión *Server* no tiene límite.

TOLERANCIA A FALLOS DE DISCO

El *driver* de disco de *Windows NT Server* (*FTDISK.SYS*) al contrario que el de la versión *Work Station*, posee tolerancia a los fallos. De hecho es capaz de duplicar de forma exacta un disco (*disk mirroring*), con una redundancia de datos del 100% de forma que si un disco falla se extraen los datos del otro. Del mismo modo también existe la posibilidad de que cada disco tenga un controlador distinto (*disk duplexing*), por si éste falla.

SERVICIOS DE RED

En lo referente a los servicios de *Red*, hay tres importantes mejoras respecto a la versión *Work Station*, por una parte se ha incluido un servicio de servidor *DHCP* (*Dynamic Host Configuration Protocol* o Protocolo de configuración dinámica del *Host*) que permite asignar direcciones *IP*.

Por otra parte la versión *Server* también posee el servicio *WINS*

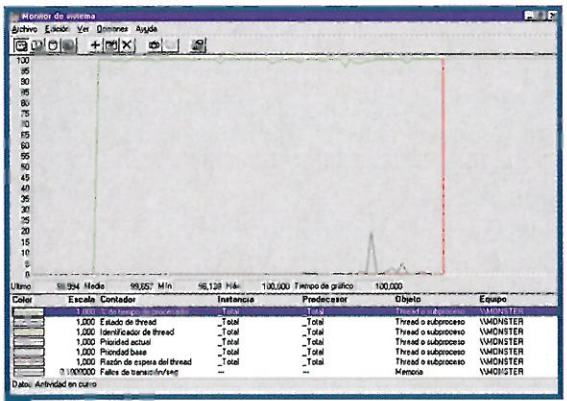


Figura 4.- Monitor del sistema con el control de los diversos hilos.

Tabla 1.- Novedades de Windows NT 4

Interfaz de usuario tipo *Windows* 95.
API de telefonía (conocida como *TAPI*) y *unimodem*.
Soporte de cliente y archivo para redes *Netware* 4.
Servicios *web* punto a punto.
Aplicaciones distribuidas para *Internet* (Arquitectura *COM* (*Commom Object Models*) o controles *ActiveX*).
Soporte de las *API DirectDraw* y *DirectSound*.
Perfiles para distintas configuraciones de *hardware*.

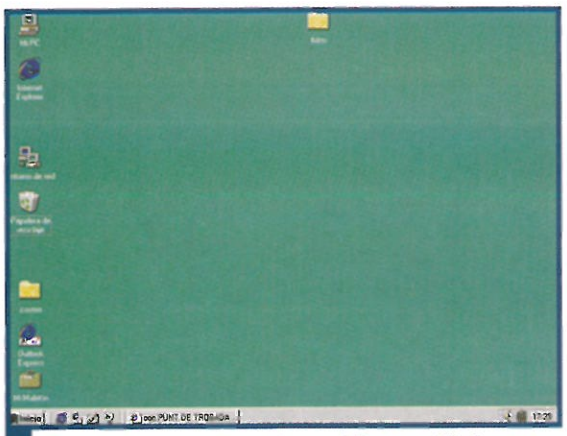


Figura 5.- Nueva interfaz gráfica en *NT 4.0* tipo *Windows 95*

(*Windows Internet Naming Service* o Servicio de Nombres de *Internet*) que nos otorga nombres dinámicos para los ordenadores en los que el servicio *DHCP* ha asignado una dirección *IP*. Y el último servicio añadido es el de *DNS* (*Domain Name System* o Sistema de Nombres de Dominio) que efectúa una correspondencia entre nombres estáticos y direcciones IP dentro de una red *TCP/IP*.

SERVICIOS RAS

Aunque los servicios de *RAS* de *Windows NT* son prácticamente iguales, el servicio en la versión *WorkStation* tan sólo permite una llamada entrante y la versión *Server* permite hasta 256.

INTEGRACIÓN CON NETWARE

Otro de los aspectos implementados en *NT Server* se refiere a ciertas herramientas que permiten la integración con redes *Novell*. En esta versión se incluye una utilidad para migrar una *Red Novell* hacía *Windows NT*, realizando automáticamente diversos pasos como por ejemplo los ficheros y directorios, permisos, seguridad, etc.

También se ha incluido una pasarela o *Gateway* de forma que los clientes de sesiones *Microsoft* puedan acceder a los servidores de redes *Novell* sin tener que instalar ningún *software* cliente de dicha *Red*.

CONTROLADOR DE DOMINIO

Una de las opciones más apreciadas de la versión *Server* es que puede actuar como un controlador de dominio. Así podemos guardar nuestro perfil en el servidor y éste permanecerá intacto, independientemente del sitio donde nos conectemos.

Dicha opción también permite que cuando los usuarios entren desde algún sistema *(Windows 3.x, Windows 9x o Windows NT)* se puedan ejecutar automáticamente programas u órdenes sobre el sistema cliente que ha entrado. Además con entrar tan sólo una vez en el sistema tendremos disponible todos los recursos de éste, sin necesidad de más *passwords*.

SERVICIOS PARA MACINTOSH

NT Server puede actuar como un servidor de ficheros e impresora para *Mac*, así estos ordenadores pueden ver la partición *NTFS* o realizar impresiones desde el servidor.

También desde un ordenador con *NT Server* podemos reinicializar un ordenador con *MS-DOS, Windows 3.1, Windows 95* (aunque dichos sistemas deben poseer una tarjeta de *Red* con un *chip* de *RPL* (Carga de programas remota)

HERRAMIENTAS ADICIONALES

Existe también una herramienta denominada *Administrador de clientes de Red* que se utiliza para generar un disco de arranque, para cuando el ordenador sea reinicializado. Esto tiene gran utilidad para instalar de forma automática el *software* cliente de *Microsoft* desde la *Red*.

Asimismo, se ha incluido en la versión *Server* una herramienta que permite monitorizar la red, para ver el tráfico que circula por ella. Esta herramienta permite desemsamblar los distintos paquetes en circulación y así aislar las zonas donde aparezca un problema.

Como es lógico, una de las grandes diferencias entre las dos versiones es la optimización interna de código para adaptarlo a sus respectivas tareas. Entre estas diferencias destacan las reflejadas en la Tabla 2.

INTERNET INFORMATION SERVER

Una de las diferencias notables entre *Windows NT Server* y *Windows WorkStation* es la inclusión en el primero del programa *Internet Information Server* 2.0 (inicialmente con la versión 4 de *Windows NT* ya apareció este programa). Recientemente *Microsoft* ha lanzado *Windows NT Server Option Pack 4*, una ampliación que incorpora una nueva versión de este producto (4.0) y que analizaremos de forma algo más detallada en páginas posteriores.

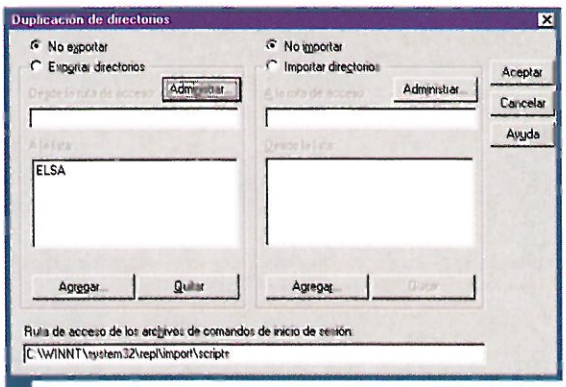


Figura 6.- Herramientas de duplicación de directorios de *NT*.

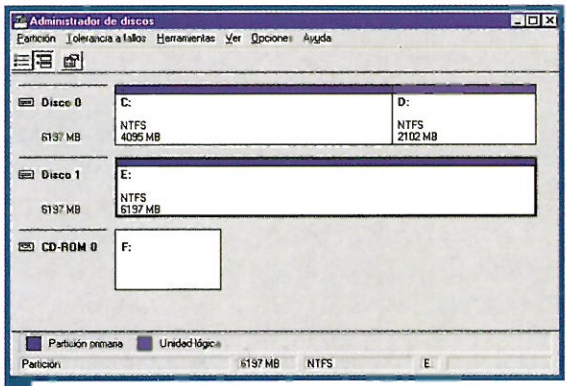


Figura 7.- Administrador de discos, en donde vemos las opciones de tolerancia de fallos.

OPTION PACK 4

Microsoft Windows NT Option Pack es un conjunto de aplicaciones/utilidades de *Web* y servicios para desarrollar la nueva generación de aplicaciones distribuidas sobre redes. El paquete incluye: *Internet Information Server 4.0, Microsoft Transaction Server 2.0. Microsoft Queue Server 1.0. Internet Connection Services para Microsoft RAS.*

Sus mejoras están orientadas a la administración y diagnóstico en:

- Configuración horaria.
- Herramientas de escritorio.
- Diagnósticos (monitorización....).
- Discos (acceso a bajo nivel, tolerancia a fallos...).
- Ficheros (comparación...).
- *Setup* y *Shutdown* de servidor.
- Documentación.

INTERNET INFORMATION SERVER 4.0

Veremos un resumen de las principales características que incluye *Internet Information Server 4.0*, entre las que destacan:

- Protección contra errores.
- Compatibilidad con *Java*.
- Múltiples sitios *Web* en una sola *IP*.
- Estándares de *Internet*.
- Administración y control del contenido.
- Páginas Activas de Servidor (ASP).
- Cola de mensajes integrada
- Administración automatizada

Veámos ahora cada característica un poco más detenidamente.

PROTECCIÓN CONTRA ERRORES

Es una de sus principales características. Esta protección permite a los clientes ejecutar diversas aplicaciones *Web* en el servidor de una manera fiable y segura. En el caso de que en un momento dado una aplicación produzca un error, el servidor *Web* y las demás aplicaciones continuarán ejecutándose como si no hubiese ocurrido nada, y la aplicación que ha producido el fallo se reiniciará con la siguiente petición.

COMPATIBILIDAD CON JAVA

IIS también posee un conjunto de clases del lenguaje *Java* y la máquina virtual de *Java 1.1* de *Microsoft*. Estas características sirven para crear componentes que se ejecutan desde el servidor.

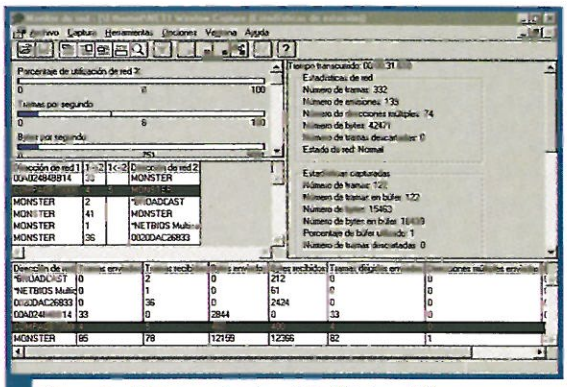


Figura 8.- Administrador de red de *Windows NT*.

MÚLTIPLES SITIOS WEB EN UNA IP

Otra de las características de *Internet Information Server* es que, en una misma dirección *IP*, admite múltiples sitios *Web*, así como la utilización de distintos operadores de los sitios, ajustar el ancho de banda dependiendo de cada uno y una buena administración totalmente personalizable que permite a las diferentes organizaciones alojar múltiples *intranets* o múltiples sitios de *Internet*.

ESTÁNDARES DE INTERNET

También se han incluido los últimos estándares de *Internet*, incluyendo *HTTP 1.1*, creado para ofrecer un mayor rendimiento, a la vez que una mayor velocidad en la conexión a la red.

ADMINISTRACIÓN Y CONTROL DEL CONTENIDO

Los administradores de una página *Web* pueden configurar ahora el servidor en distintos niveles: por sitio, por directorio virtual, por subdirectorio, por archivo, etc. Ello permite una gran flexibilidad en las configuraciones del servidor. También se pueden crear mensajes de error personalizados.

ACTIVE SERVER PAGES

También incluido en *IIS 4.0* encontramos el producto *Transactional Active Server Pages*, dicho módulo permite a las aplicaciones que poseen secuencias de comandos y arquitectura de componentes, realizar acciones de

Tabla 2: Optimización del código entre la dos versiones de Windows NT

Propiedades	Server	WorkStation
Escritura en la memoria de cache	Más grande y se limpia con menor frecuencia	Menor y actualizada en intervalos menores.
Paginación del código	El código casi no se pagina, para así permanecer en memoria y responder mejor a las demandas.	Paginación adicional de código (se requiere menos memoria física)
Carga de una máquina virtual DOS	Se carga bajo demanda	Se carga al principio.
Hilos del sistema	Mayor número de hilos	Menor número, dejando así más recursos para las aplicaciones.
Optimización de servicios fichero/impresora.	Permite escoger entre optimizar el sistema para aplicaciones de servidor o para aplicaciones de ficheros/impresiones.	No permite la optimización.

forma simultánea o anularlas simultáneamente (las transacciones). Este hecho es un paso absolutamente necesario e imprescindible cuando se trabaja con aplicaciones de bases de datos.

COLA DE MENSAJES INTEGRADA

La cola de mensajes se ha integrado dentro de *IIS 4.0,* de esta forma las aplicaciones pueden enviar y recibir mensajes fácilmente y de forma fiable a través de una red, incluso cuando la red o parte de la aplicación no estén disponibles en ese momento.

ADMINISTRACIÓN AUTOMATIZADA

También podemos escribir secuencias de comandos y ejecutarlas desde la línea de comandos para así automatizar diferentes tareas de administración. Se han incluido también conjuntos de ejemplos de secuencias de comandos.

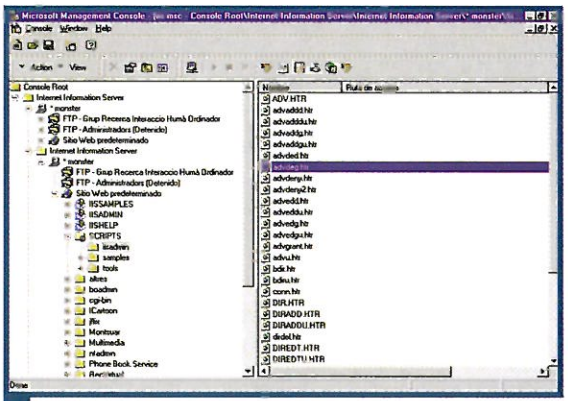


Figura 9.- Aspecto que presenta la consola gráfica de *IIS*.

Además, dentro de *IIS 4.0* tenemos también una solución integrada para edepurar las distintas secuencias de comando y aplicaciones *ASP* (*Active Server Pages* o Páginas Activas de Servidor).

Vamos ahora una serie de características interesantes, en las que se ha hecho especial hincapié en esta nueva versión:

SERVIDOR WEB MÁS FÁCIL DE INSTALAR Y CONFIGURAR

Con esta nueva versión se ha mejorado el asistente de descarga de los componentes que se deseen instalar. También existe una nueva capacidad de instalación, llamada *instalación desatendida*, que permite instalar *IIS* en varios servidores sin tener que supervisar las instalaciones.

HERRAMIENTAS FLEXIBLES DE ADMINISTRACIÓN

Las nuevas herramientas de administración del servidor incluyen una consola integrada de administración basada en *Windows*, así como sentencias de comandos ejecutables desde la línea de comandos e interfaces programables para que el usuario pueda crear herramientas de administración personalizadas.

SEGURIDAD

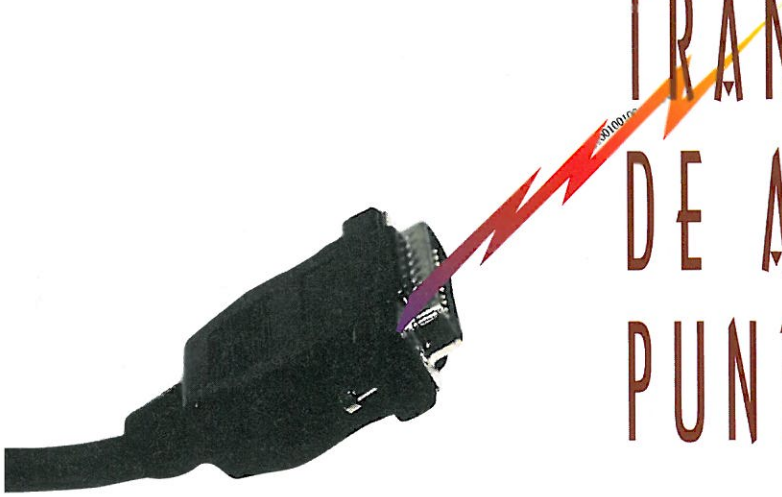
En esta versión se ha añadido un servidor de certificados digitales integrado en el modelo de *Windows NT Server*. Dicho servidor permite a los usuarios emitir certificados digitales estándar de *Internet* del tipo *X.509*.

MOTOR DE BÚSQUEDA

Ahora, *IIS 4.0* incluye capacidades de búsqueda integradas, que permiten que los usuarios puedan crear formularios de búsqueda personalizados con *ASP*, así como usar objetos *ActiveX* y realizar consultas *SQL* (*Structured Query Language* o Lenguaje Estructurado de Consultas) para buscar la información que se desee en Web.

Hasta aquí hemos visto un resumen de las características de *Windows NT* e *Internet Information Server*. En el próximo artículo se comentarán de forma detallada cada una de estas características, describiendo, además, la configuración de los servicios *Web* y *FTP*.

TRANSFERENCIA DE ARCHIVOS PUNTO A PUNTO (III)



Enrique de la Lastra (enriquelastra@hotmail.com)

Una vez que concluyamos el desarrollo del componente que se encarga de la transmisión de datos, habrá que implementar un programa que compruebe su correcto funcionamiento. La aplicación que hemos elegido es una aplicación que permite transmitir mensajes de texto entre dos ordenadores, un *chat*.

INTRODUCCIÓN

En el artículo anterior comenzamos el desarrollo de un componente en *Delphi* cuyas funciones básicas consistían en abrir y cerrar el puerto serie y proveer los mecanismos necesarios para transmitir y recibir datos a través de él. Además, y aunque esto es una característica del diseño orientado a objetos, debía encapsular las funciones de la *API* (*Application Programming Interface*) de *Windows* y ocultarlas al exterior.

En las siguientes líneas terminaremos este componente e implementaremos un programa de prueba que permita realizar un *chat* entre dos ordenadores conectados entre sí a través de sus puertos serie. Mediante este programa de prueba podremos enviar cadenas de texto en los dos sentidos de la comunicación y mostraremos de forma secuencial tanto los mensajes enviados como los recibidos.

El componente no verifica, por diseño, si se han producido errores de transmisión

Esta aplicación, que hace de programa de prueba, tendrá una serie de limitaciones, que se solventarán de forma transparente mediante el desarrollo de los siguientes niveles de la jerarquía de clases (tal y como explicamos en el primer artículo de la serie). Entre las limitaciones se encuentra la incapacidad de enviar cadenas de texto de más de 255 caracteres, así como la no verificación si la cadena es de una longitud mayor que ese tamaño. Además no se comprueba (ya que el componente no proporciona ningún medio para hacerlo) si los mensajes de texto han llegado con errores.

Aparte de las limitaciones intrínsecas descritas, el programa de prueba tendrá una funcionalidad limitada intencionadamente. Por ejemplo, no se podrán enviar archivos de un ordenador a otro, ya que para ello es necesario que el componente se comporte de forma fiable

tanto en la transmisión como en la recepción de los datos, lo cual no se cumple por definición de diseño.

AVISO DE LA RECEPCIÓN DE DATOS

El componente, según vimos, emplea un *Timer* para sondear cada cierto tiempo el puerto serie y comprobar si han llegado nuevos datos al *buffer* de entrada.

En el momento en que eso ocurre, disparamos un evento personalizado en el que, no sólo avisamos de la llegada de datos, sino que a través de un par de parámetros, pasamos un puntero al *buffer* de entrada y una variable con el número de *bytes* que hay en este *buffer*.

El evento OnDataReceived avisa de la llegada de datos al puerto serie

Para crear este evento, definiremos un nuevo tipo "procedimiento", es decir un tipo cuyo valor es un puntero a un procedimiento (tal y como se definen los tipos de los eventos). Los parámetros de este tipo de procedimiento, serán además de la propia clase, un puntero al *buffer* de entrada y un entero que almacene el número de datos recibidos en el *buffer*:

```
type
  TOnDataReceived = procedure (
        Sender: TObject;
        Datos: pointer;
        NumDatos: integer)
  of object
```

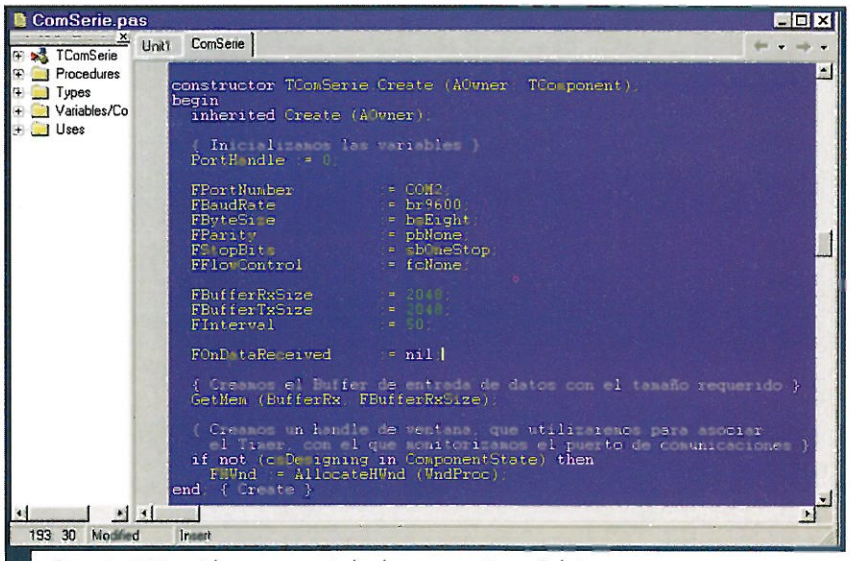


Figura 1.- Código del constructor de la clase, mostrado en *Delphi*.

Como queremos que este evento pueda ser utilizado tanto por el nivel superior en la jerarquía de clases, como por el usuario final desde el *Object Inspector*, crearemos en el componente una propiedad *published* para que sea visible en ambos casos:

```
TComSerie = class(TComponent)
protected
 ...
 FOnDataReceived: TOnDataRecei-
    ved;

published
 ...
 property
  OnDataReceived: TOnDataReceived
            read  FOnDataRecei-
    ved
           write  FOnDataRecei-
    ved;
```

Y el disparo del evento lo realizaremos cuando hayan llegado datos al puerto serie. Tal y como vimos, esto se realiza en el procedimiento *WndProc* - encargado de procesar todos los mensajes que llegan de *Windows* - cuando llega el evento *WM_TIMER* y se cumple que el número de datos leídos es distinto de **0**.

Por último habrá que iniciar a **nil** el valor del evento *FOnDataReceived* en el constructor de la clase. Podemos ver la línea necesaria en la Figura 1, donde se muestra el código del método *Create*.

FRECUENCIA DE CHEQUEO

Dado que para comprobar la llegada de nuevos datos utilizamos un *Timer*, una propiedad que sería deseable poder modificar sería la frecuencia con la que chequeamos el puerto serie. Habrá aplicaciones que requieran un chequeo constante del puerto - por ejemplo todas aquéllas que operen en tiempo real - mientras que otras no tendrán necesidad de comprobar el puerto tan a menudo. Para arreglar esto, haremos público el intervalo de comprobación del puerto serie, de forma que sea modificable en diseño y en ejecución.

La variable, que denominaremos **Interval**, almacenará el tiempo en milisegundos entre dos mensajes

WM_TIMER, y la utilizaremos en todas las llamadas a la función *SetTimer*. Las declaraciones que debemos realizar en la clase serán:

```
TComSerie = class(TComponent)
protected
  ...
  FInterval: word;
  procedure SetInterval
            (AValue: word);

published
  ...
  property Interval: word
     read  FInterval
    write  SetInterval;
```

El método *SetInterval* además de actualizar el valor de la propiedad *FInterval* se encargará de establecer el nuevo intervalo de comprobación del puerto, en caso de que éste se encuentre abierto:

```
procedure TComSerie.SetInterval
          (AValue: word);
begin
 if AValue <> FInterval then
    begin
   FInterval := AValue;
   if PortHandle > 0 then begin
     KillTimer(FHWnd, 1);
     SetTimer(FHWnd, 1234,
              FInterval, nil);
   end; { if }
 end; { if }
end; { SetInterval }
```

BUFFER DE ENTRADA

En el evento *OnDataReceived* definimos como parámetro un puntero al *buffer* de entrada. En realidad, ese *buffer* de entrada no existe como tal asociado al puerto serie, sino que debemos crearlo nosotros y asignarle los datos que van llegando al puerto abierto.

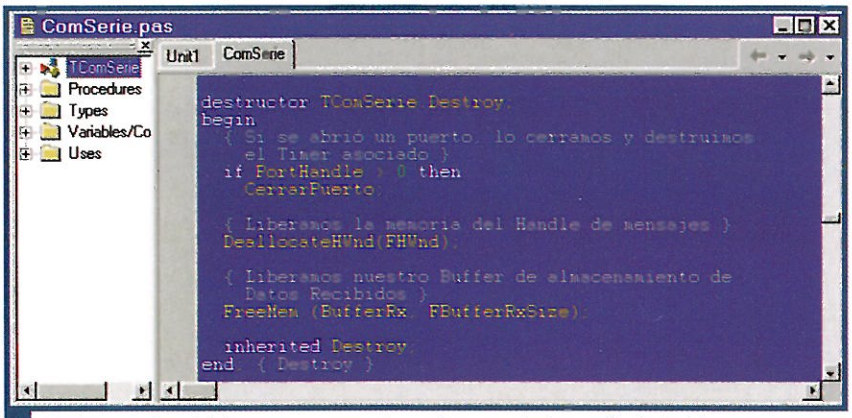


Figura 2.- Código del destructor, mostrado en el entorno de desarrollo de *Delphi*.

Por tanto, declararemos una nueva variable, *BufferRx*, que se creará con el tamaño definido a través de la propiedad *BufferRxSize*. Esta variable será privada, pues queremos que el *buffer* de entrada sólo sea accesible a través del evento *OnDataReceived*.

```
TComSerie = class(TComponent)
private
  ...
  BufferRx: pointer;
```

Al actualizar la propiedad *BufferRxSize* deberemos cambiar también el tamaño de *BufferRx*, por lo que habrá que modificar el método *SetBufferRxSize*:

```
procedure TComSerie.SetBufferRxSi-
    ze
          (AValue: word);
begin
 if (PortHandle > 0) then begin
  FreeMem(BufferRx,FBufferRxSi-
    ze);
  FBufferRxSize := AValue;
  GetMem (BufferRx,FBufferRxSi-
    ze);
 end; { if }
end; { SetInterval }
```

TIMEOUTS

El puerto serie tiene una serie de *Timeouts* que afectan a las operaciones de lectura y escritura, llegando a finalizarlas incluso antes de que se hayan terminado de leer o escribir todos los datos solicitados.

Los Timeouts de escritura evitan el bloqueo por control de flujo

Los *Timeouts* de escritura son útiles cuando la transmisión está bloqueada debido al control flujo. Por su parte, los de lectura son útiles, además, cuando hay una pausa en la recepción de datos, lo que produciría a su vez una pausa en el programa, dentro de la llamada a la función de lectura del puerto serie.

Aunque estos *Timeouts* toman unos valores por defecto al abrir el puerto es necesario fijarlos de acuerdo con nuestros requisitos personales. Para ello disponemos de dos funciones de la *API* de *Windows*: *GetCommTimeouts* y *SetCommTimeouts*, que permiten, respectivamente, leer y escribir los valores de esos *Timeouts*. Ambas funciones tienen como primer parámetro el *Handle* al puerto, y como segundo parámetro una estructura de tipo *TCOMMTIMEOUTS*. La estructura *TCOMMTIMEOUTS* está compuesta de las siguientes variables:

- *ReadIntervalTimeout*: Máximo tiempo, en milisegundos permitido entre la llegada al puerto de dos caracteres. Si

el intervalo entre dos caracteres es mayor que este valor, la llamada a *ReadFile* finaliza y se devuelven todos los datos almacenados.

De esta manera, se garantiza que si pasa mucho tiempo entre la recepción de dos caracteres, no se detenga la ejecución dentro de la llamada a la función *ReadFile*. Si se fija a **0**, significa que no se utiliza.

- *ReadTotalTimeoutMultiplier*: Especifica el multiplicador (en milisegundos) utilizado para calcular el *Timeout* total para las operaciones de lectura. En cada operación se multiplica este valor por el número de *bytes* que se quieren leer del puerto, consiguiendo de esta forma que el *Timeout* sea proporcional a la cantidad de datos esperados.

Así, si se fija a esta variable un valor **X**, las operaciones de lectura en el puerto terminarán cuando hayan transcurrido **X** milisegundos desde que se recibió el primer *byte*, o cuando el *buffer* de entrada esté lleno.

Los Timeouts de lectura permiten continuar la ejecución aunque haya una pausa en la recepción

- *ReadTotalTimeoutConstant*: Constante (en milisegundos) utilizada para calcular el *Timeout* total para las operaciones de lectura. En cada operación de lectura se suma este valor al resultado de la multiplicación anterior. Fijando esta variable (por ejemplo con un valor **Y**) la operación de lectura finaliza cuando han pasado **Y** milisegundos o el *buffer* de entrada se ha llenado. Si tanto a esta variable como a la anterior se les asigna un valor igual a **0**, no se utiliza *Timeout* de lectura, y por tanto estas operaciones terminan cuando el *buffer* de entrada está lleno.

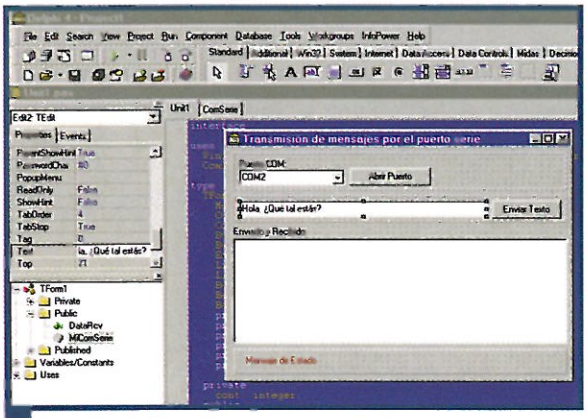


Figura 3.- Aplicación de prueba en modo diseño.

- *WriteTotalTimeoutMultiplier*: Es igual que *ReadTotalTimeoutMultiplier* pero para operaciones de escritura.

- *WriteTotalTimeoutConstant*: El significado de esta variable es parecido al de *ReadTotalTimeoutConstant* (para operaciones de escritura). La diferencia está en que aquí se aplica para el tiempo transcurrido desde el primer *byte* recibido y no para el tiempo transcurrido entre la llegada de dos *bytes* cualesquiera como ocurría con *ReadTotalTimeoutConstant*.

En nuestro caso, utilizaremos *Timeouts* tanto en lectura como en escritura. Los valores que emplearemos serán: **2** milisegundos para los multiplicadores y **1** milisegundo para las constantes. El código necesario para fijar estos *Timeouts*, que introduciremos dentro de la función *AbrirPuerto*, será el siguiente:

```
With tms do begin
  ReadIntervalTimeout:= 1;
  ReadTotalTimeoutMultiplier :=
    2;
  ReadTotalTimeoutConstant    :=
    1;
  WriteTotalTimeoutMultiplier:=
    2;
  WriteTotalTimeoutConstant   :=
    1;
end;
SetCommTimeouts (PortHandle,
    tms);
```

CERRAR EL PUERTO

Ya tenemos todas las funciones necesarias para abrir el puerto y transmitir y recibir datos a través de él. Pero, ¿qué ocurre si deseamos abrir otro puerto serie distinto del que está abierto en un momento dado?. Como en el método *SetPortNumber* se comprueba primero si el puerto está cerrado para establecer un nuevo valor, no es posible cambiar de puerto una vez abierto uno.

Definimos un procedimiento para cerrar el puerto que se encuentre abierto

Para evitar esto, definimos un procedimiento *CerrarPuerto* que se encargue de esta tarea. En este

procedimiento deberemos vaciar los *buffers* asociados al antiguo, terminar el *Timer*, cerrar el *Handle* asociado y actualizar la variable *PortHandle* a **0**, valor que indica que no hay un puerto abierto.

Por tanto el código de este procedimiento quedará como sigue:

```
procedure TComSerie.CerrarPuerto;
begin
  if PortHandle > 0 then begin
    FlushBufferRx;
    FlushBufferTx;
    KillTimer (FHWnd, 1);
    CloseHandle (PortHandle);
    PortHandle := 0;
  end;
end; { CerrarPuerto }
```

Además por el hecho de añadir esta función, la función *AbrirPuerto* cambia un poco. La razón es que para abrir un puerto hay que cerrar el que estuviese abierto. Esto significa que *AbrirPuerto* debe devolver inicialmente **False** si ya hay un puerto abierto, en lugar de **True** como devolvía antes.

CONSTRUCTOR Y DESTRUCTOR

Sólo queda determinar las variables que hay que iniciar y destruir en la creación y destrucción de la clase.

En el constructor daremos valores a todas las propiedades y crearemos el *buffer* de entrada de datos y el *Handle* a la ventana que recibe los mensajes de *Windows*. Se puede observar el código en la figura 1.

En cuanto al destructor, deberá destruir las anteriores variables además de encargarse de cerrar el puerto serie, si hay alguno abierto. Su código se muestra en la figura 2.

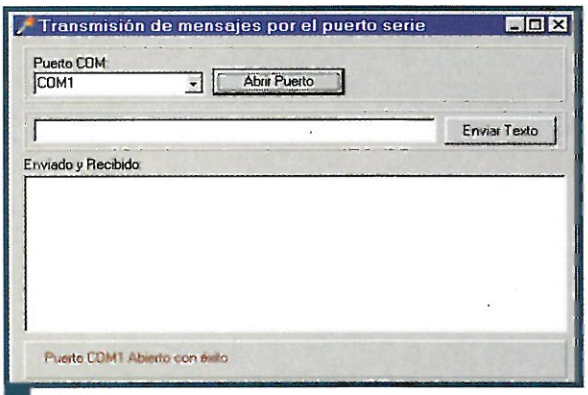


Figura 4.- Imagen de la pantalla de la aplicación después de abrir el puerto *COM1*.

Con esto queda terminado el componente *TComSerie*. El código completo del mismo se acompaña en el *CD-ROM* de la revista junto con el código del programa de prueba que veremos a continuación.

PROGRAMA DE PRUEBA

Como hemos comentado al principio del artículo, vamos a desarrollar un programa de prueba que permita verificar el correcto funcionamiento de nuestro componente. El programa en cuestión permitirá enviar cadenas de texto entre dos *PCs* que estén conectados mediante sus puertos serie.

El programa de prueba hará de chat a través del puerto serie

Para ello, y haciendo uso del componente que acabamos de crear, deberá permitir cambiar el puerto serie que queremos abrir (se realizará mediante un *Combo Box*, el cual permite delimitar los valores posibles: *COM1* y *COM2*). Deberá ofrecer además un cuadro de texto con el que podamos teclear el mensaje que enviar y un *memo* en el que mostrará los mensajes enviados y recibidos. El aspecto que va a tener esta aplicación se muestra en la figura 3.

En esta figura 3 podemos apreciar que se ha definido una "línea de estado", que servirá para informar tanto de los errores que se hayan producido como de los distintos sucesos que vayan ocurriendo en el transcurso de una comunicación.

Para verificar el código del componente sin necesidad de tener que añadirlo a la *VCL* (*Visual Component Library, Biblioteca de Componentes Visuales*), crearemos el mismo mediante una llamada directa al constructor, en el método *FormCreate* de la aplicación de prueba.

El método *FormCreate* (ver listado), además de crear el componente, asignará los valores por defecto del puerto serie y del manejador del evento *OnDataReceived*. En nuestro caso, el puerto por defecto será el COM1.

Al elegir el puerto serie en el *ComboBox*, cerraremos el puerto serie, actualizaremos la variable *PortNumber* y abriremos el puerto (llamando al método *Button1Click*):

```
procedure TForm1.ComboBox1Change
                (Sender: TObject);
begin
  MiComSerie.CerrarPuerto;
  MiComSerie.PortNumber :=
TPortNumber(ComboBox1.ItemIndex);
  Button1Click(nil);
end;
```

Listado 1: Código del constructor y destructor de la aplicación de prueba, así como del manejador del evento OnDataReceived.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  MiComSerie := TComSerie.Create (Self);
  MiComSerie.PortNumber := TPortNumber(0);
  MiComSerie.OnDataReceived := DataRcv;
  ComboBox1.ItemIndex := ord (MiComSerie.PortNumber);
  cont:= 0;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  MiComSerie.Free;
end;

procedure TForm1.DataRcv (Sender: TObject; Datos: pointer; NumDatos:
integer);
var
  pRecibido: PChar;
  Recibido: string;
begin
  pRecibido := Datos;
  Recibido := '';
  while NumDatos > 0 do begin
    Recibido := Recibido + pRecibido^;
    inc(pRecibido);
    dec(NumDatos);
  end;
  inc(cont);
  Memo1.Lines.Add (IntToStr(cont) + '. Recibido: ' + Recibido);
  Label2.Caption := 'Mensaje Recibido';
end;
```

El método *Button1Click*, se encargará de abrir el puerto serie y mostrar en la "línea de estado" el resultado positivo o negativo.

```
procedure TForm1.Button1Click
            (Sender: TObject);
var NumPort: char;
begin
  NumPort := chr(ord(
MiComSerie.PortNumber)+ord('1'));
  if MiComSerie.AbrirPuerto then
    Label2.Caption := 'Puerto
    COM'
    +NumPort+ ' Abierto con
    éxito'
  else
    Label2.Caption := 'Error al
    abrir el puerto COM' +Num-
    Port;
end;
```

En la figura 4 se puede observar la aplicación, en tiempo de ejecución, después de haberse abierto el puerto **COM1**. Al pulsar el botón *Enviar Texto* (*Button2*) se realiza una llamada a la función *EnviarDatos* del componente *TComSerie*, y se indica el resultado en la línea de estado. Además, se muestra en el memo el mensaje enviado (ver figura 5).

Por último, el manejador del evento *OnDataReceived* lo denominaremos *DataRcv* (Listado 1). Este método lo único que hace es recoger los datos que llegan en forma de puntero al *buffer* de entrada y los muestra en el memo *Enviado y Recibido*.

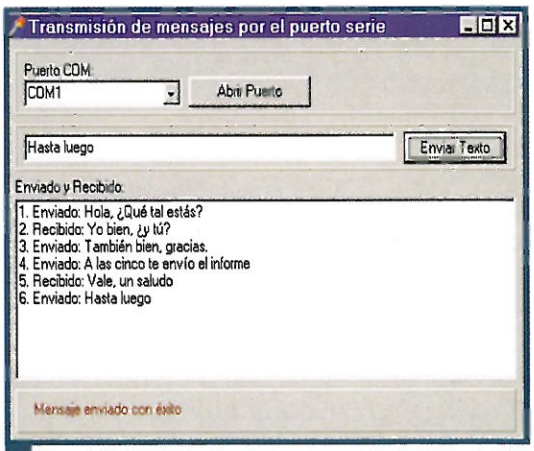
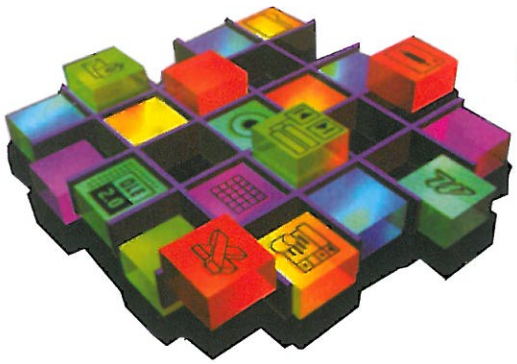


Figura 5.- Aplicación de *chat* después de haber enviado el mensaje numerado como 6.

CONCLUSIÓN

En los tres artículos escritos de la serie, hemos implementado un componente que se encarga de la transmisión y recepción de información a través del puerto serie. Este componente, que funcionalmente implementa el nivel 1 de la *OSI*, servirá como base para el desarrollo de otros dos componentes heredados, que su vez implementarán otros niveles de la arquitectura *OSI*.

Como prueba de este componente, hemos desarrollado una pequeña aplicación que hace uso del mismo para transmitir mensajes de texto entre dos ordenadores conectados a través del puerto serie.



Suite para Internet (II)
El correo electrónico (I)

Jordi Agost Moré *(agost@eup.udl.es)*

¿Cómo podemos añadir funcionalidades de correo electrónico a nuestra aplicación? ¿Cuáles son los pasos que debemos seguir? ¿Cómo lo podemos realizar con el menor número de controles posible?

En el anterior artículo de esta serie vimos cómo incorporar un navegador, con todas las posibilidades que éste ofrece a nuestro proyecto. Ahora vamos a ver cómo hacer lo propio con el que quizás constituya el servicio de mayor uso de *Internet* hoy en día. En definitiva nos referimos a la utilización de correo electrónico en nuestra aplicación.

ABRIR EL EDITOR DE MAIL DIRECTAMENTE

Algunas veces en nuestra típica caja de diálogo **Acerca de...** deseamos incluir un atajo con el fin que quien lo desee pueda enviarnos un mensaje correo electrónico. En el artículo anterior vimos cómo hacer lo mismo pero con una dirección *web*. Uno de los métodos más fáciles para realizar esta tarea consiste en invocar al programa *Start.exe*. El argumento de dicho programa sería equivalente a una supuesta cadena que podríamos escribir en el submenú **Ejecutar** del menú Inicio de *Windows 95/98* (figuras 1 y 2). Si en dicho menú añadimos por ejemplo la siguiente línea:

```
www.pruebasoloprogramadores.net
```

Conseguiremos que el explorador por defecto, se abra y que vaya a la dirección que hemos especificado en la caja de texto de la caja de diálogo **Ejecutar.**

Ahora bien si insertamos la siguiente línea:

```
mailto:solop@solop.net
```

Entonces lo que sucedería es que se abriría por defecto el

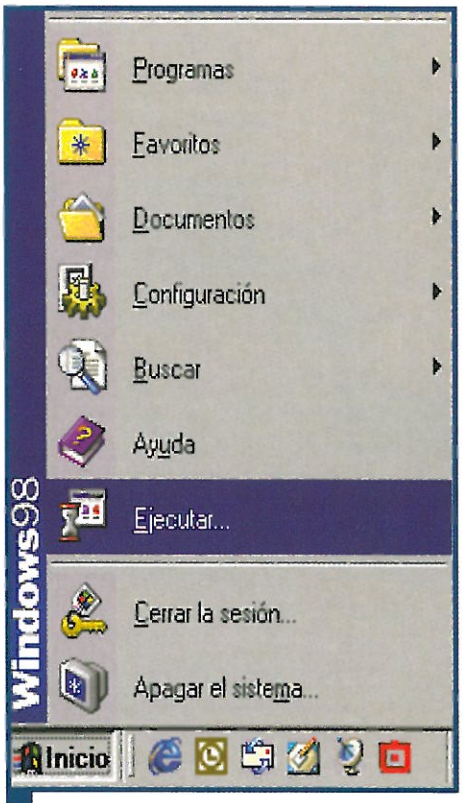


Figura 1.- Menú inicio de Windows 95/98 con el menú Ejecutar.

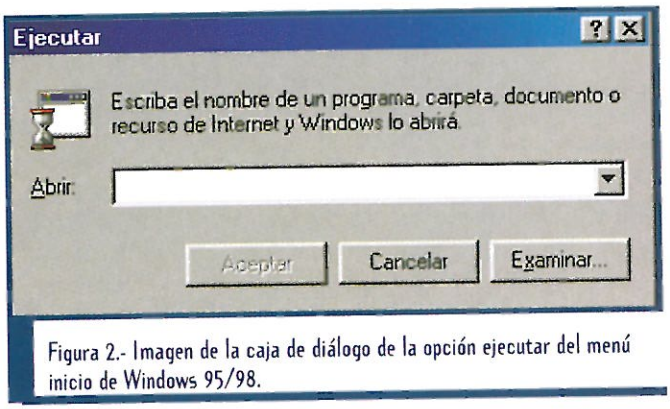


Figura 2.- Imagen de la caja de diálogo de la opción ejecutar del menú inicio de Windows 95/98.

programa de correo electrónico que tengamos instalado y en la dirección aparecería la que hemos escrito.

Una de las formas más fáciles de enviar correo electrónico es con el programa del sistema Start.exe

Todo lo que hemos dicho con anterioridad lo podemos realizar también desde el interior de nuestro programa invocando al programa **"Start.exe"** a través de la sentencia *Shell*, que nos sirve para ejecutar procesos externos, así las sentencias anteriores pasarían a convertirse en:

```
Shell "Start.exe " & "www.prueba-
    soloprogramadores.net", 0
```

Y en:

```
Shell Start.exe " & "
    mailto:solop@solop.net", 0
```

De todos modos, todavía podemos complicar un poco más el envío de nuestro *e-mail*. Así podríamos hacer que al mismo tiempo se cumplimentara también la línea de *Subject* del mismo, del siguiente modo:

```
Shell "Start.exe " & "
    mailto:solop@solop.net mail-
    to:lc-?Subject=Consultas", 0
```

Y refinando un poco más el proceso anterior también podríamos adjuntar en nuestro correo electrónico el cuerpo del mismo. Para ello tan sólo deberíamos añadir detrás del elemento *Subject* otro elemento llamado *Body* que sea equivalente a una cadena que constituirá el cuerpo de nuestro mensaje, de la siguiente forma:

```
Shell "Start.exe " & "
    mailto:solop@solop.net mail-
    to:lc-
    ?Subject=Consultas?Body=Cuerpo
    Mail", 0
```

El uso del programa Start tan sólo es apto para Windows 9x

Es necesario resaltar que todo el código mencionado tan sólo funcionará *en Windows 9x*, ya que como sabe *Windows 3.x* no dispone del menú *Inicio*.

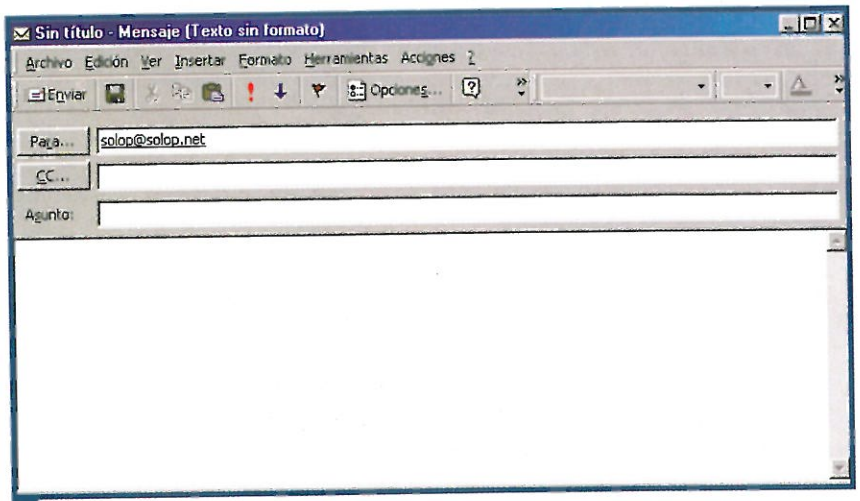


Figura 3.- Resultado de llamar el programa Start con direcciones de e-mail.

ENVÍO DE CORREO ELECTRÓNICO MEDIANTE EL CONTROL WINSOCK

EL CONTROL WINSOCK

Otro de los medios para enviar un *e-mail* directamente desde la aplicación, sin utilizar programas propios del sistema, es a través del control *Winsock*, que actuará sobre el puerto correspondiente al correo electrónico.

Básicamente su función consiste en realizar una interfaz entre la *API* de nivel inferior y nuestro programa, para conseguir de este modo facilitarnos la programación de la comunicación entre diferentes equipos.

PROTOCOLOS DE CONEXIÓN

El control *Winsock* de *Visual Basic* permite acceder a los servicios de red *TCP (Transfer Control Protocol* o Protocolo de control de transferencia*)* y *UDP*

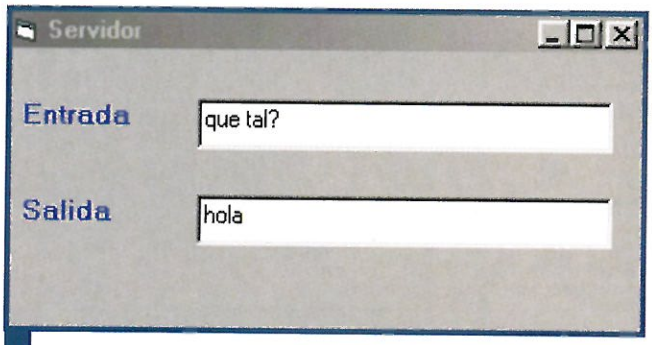


Figura 7.- Aspecto de la aplicación servidor.

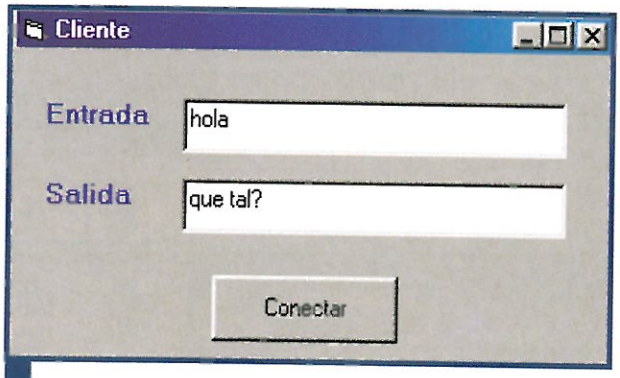


Figura 8.- Aspecto de la aplicación cliente.

En primer lugar deberemos crear un proyecto de tipo *EXE estándar*, asignado al formulario que aparece por defecto el nombre **frmServidor** (ver figura 7).

Añadiremos un control *Winsock* al formulario y lo nombraremos como **WinsockS**. Seguidamente añadiremos dos controles de caja de texto, que denominaremos **txt Enviar** y **txtMensaje** respectivamente, evitando confusiones.

Para crear una aplicación TCP primero deberemos establecer la comunicación entre dos equipos

En el evento *Form_Load* estableceremos la propiedad *LocalPort* en un entero, y haremos que nuestra aplicación servidor (el formulario que estamos creando) escuche el puerto, para después ejecutar el formulario cliente.

```
Private Sub Form_Load()
  WinSockS.LocalPort = 1001
  WinSockS.Listen
  frmClient.
End Sub
```

Será en el evento *Connection Request* en el que determinaremos si el control tiene una conexión. De ser así la cerraremos para poder realizar nuestra propia conexión, sin que nos aparezcan errores:

```
Private Sub
    WinsockS_ConnectionRequest _
(ByVal requestID As Long)

If WinSockS.State <> sckClosed
    Then _
WinSockS.Close
WinSockS.Accept requestID
End Sub
```

En el control de caja de texto *txtEnviar*, que antes hemos añadido, siempre que escribamos algo haremos que lo escrito se envíe directamente a la aplicación cliente con el método *SendData*.

```
Private Sub txtEnviar_Change()
  WinSockS.SendData
    txtEnviar.Text
End Sub
```

En el evento ConnectionRequest determinaremos las conexiones abiertas y decidiremos si las cerramos o no

En el evento *DataArrival* del control *Winsock* declararemos una variable para los datos que entran, los recopilaremos y los mostraremos en la caja de texto:

```
Private Sub WinsockS_DataArrival
    (ByVal bytesTotal As Long)
  Dim strData As String
  WinSockS.GetData strData
  txtMensaje.Text = strData
End Sub
```

UN EJEMPLO SIMPLE DE CLIENTE TCP

Para crear la aplicación cliente añadiremos un formulario llamado **frmCliente**, en el que insertaremos un control *Winsock* que llamaremos *WinsockC*. Seguidamente ñadiremos las dos cajas de texto a las cuales nombraremos como **txtMensaje** y **txtEnviar**. Y en este formulario añadiremos por último un botón llamado **cmdConectar**.

El evento DataArrival es el encargado de recibir los datos provenientes del otro equipo en una comunicación TCP

En el código de inicio del formulario, especificaremos el nombre del ordenador remoto. Aquí el lector deberá insertar el nombre de su equipo, o en su defecto la dirección *IP* del mismo. Y posteriormente estableceremos un puerto para comunicarnos.

```
Private Sub Form_Load()
  WinSockC.RemoteHost = "MARTA"
  WinSockC.RemotePort = 1001
End Sub
```

El código del botón invocará al método *Connect* para así iniciar una determinada conexión:

```
Private Sub cmdConectar_Click()
  WinSockC.Connect
End Sub
```

Y tal y como habíamos hecho con el servidor, cada vez que haya un cambio en la caja de texto, enviaremos dicho cambio a la conexión que tengamos establecida.

```
Private Sub txtEnviar_Change()
  WinSockC.SendData
    txtEnviar.Text
End Sub
```

Análogamente al formulario anterior, recogemos los datos que recibimos y los mostramos por la caja de texto:

```
Private Sub WinsockC_DataArrival
    (ByVal bytesTotal As Long)
  Dim strData As String
  WinSockC.GetData strData
  txtMensaje.Text = strData
End Sub
```

Las rutinas que hemos visto hasta ahora constituyen una aplicación cliente/servidor sencilla, las cuales podremos probar ejecutandolo y haciendo *clic* en el botón **Conectar**. De esta forma conseguiremos que lo que se escriba en el control **txtEnviarDato** aparezca en la caja de texto **txtSalida** del otro formulario.

CONFECCIÓN DE UN PROGRAMA DE ENVÍO DE E-MAIL

El *mail* que vamos a enviar va a seguir el protocolo *SMTP (Simple Mail Transfer Protocol)*. Este protocolo es un estándar de *Internet* para enviar mensajes de correo electrónico entre programas clientes que tienen acceso a un servidor en el que se almacena el correo.

Para el envío de e-mail nos basaremos en el protocolo SMTP, un estándar de Internet

Para quien desee más información sobre dicho protocolo diremos que se encuentra especificado en el documento *RFC 821* para el estándar del mensaje y *RFC 822* para el formato del protocolo. Dichos documentos son fácilmente accesibles a través de *Internet*, y se incluyen en CD-ROM

Vamos a ver ahora el funcionamiento de los principales elementos que constituyen nuestro programa de envío de correo.

Uno de los aspectos más importantes será el evento *DataArrival* del control *Winsock*. En dicho evento deberemos ejecutar el método *GetData* del control *Winsock*, especificándole como argumento una variable de tipo *Variant*, que será la encargada de almacenar la respuesta del servidor.

```
Private Sub
    Winsock_DataArrival(ByVal
    bytesTotal As Long)

Winsock.GetData Respuesta

End Sub
```

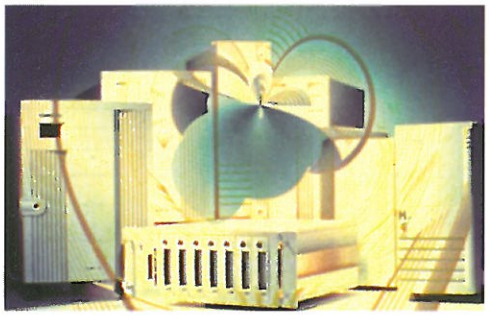
Analicemos a continuación la rutina que servirá para enviar el correo electrónico. En primer lugar deberemos observar el estado del control *Winsock*, que se realiza mediante la propiedad *State*. Dicha propiedad es de sólo lectura y no se encuentra disponible en tiempo de diseño. Podemos ver sus valores en la Tabla 1.

Antes de establecer cualquier comunicación deberemos observar el estado del control Winsock

Si el estado no es "cerrado", quiere decir que hay una comunicación abierta, con lo que deberemos esperar a que ésta se cierre para actuar nosotros. Otro aspecto importante se refiere a la asignación de la

Tabla 1. Valores de la propiedad State del control Winsock.

Constante	Valor	Descripción
SckClosed	0	Predeterminado. Cerrado
SckOpen	1	Abierto
SckListening	2	Escuchando
SckConnectionPending	3	Conexión pendiente
SckResolvingHost	4	Resolviendo host
SckHostResolved	5	Host resuelto
SckConnecting	6	Conectando
SckConnected	7	Conectado
SckClosing	8	El equipo está cerrando la conexión
SckError	9	Error



Desarrollo cliente/servidor (IV)

Javier Toledo (jtoledo@sei.es)

En el artículo anterior enfocamos el sistema cliente/servidor desde el punto de vista de un servidor de base datos *Oracle*. Ahora le toca el turno a *SQL Server*, adelantando el proceso creación de la *BBDD* que vermos en el siguiente número.

Comenzamos este artículo completando el listado de creación de nuestra base de datos *Oracle*. Con el siguiente listado creamos el usuario y las tablas necesarias para nuestra aplicación.

En nuestra base de datos almacenamos los distintos productos que tenemos disponibles en un conjunto de comercios. Nuestra sencilla base datos consta de tres tablas: una tabla de comercios, una tabla de productos y una tabla de comercios/productos mediante la que podemos saber cuáles son los productos disponibles en cada comercio y el número de los mismos en existencias.

Una vez que tengamos creado nuestro usuario nos conectaremos a la base de datos asumiendo su identidad. De este modo, cuando creemos las tablas necesarias para nuestra aplicación, dichas tablas pertenecerán a este nuevo usuario. Los objetos de este usuario necesitarán ser creados en un *tablespace*. Recordemos que actualmente nuestra base de datos posee un *tablespace*, denominado **TABLESPACECOMERCIO** para tal efecto.

```
DROP USER comercios CASCADE
/
CREATE USER comercios IDENTIFIED
     BY ORACLE default tablespace
     TABLESPACECOMERCIO temporary
     tablespace TEMPORARY;

ALTER USER comercios QUOTA UNLI-
     MITED ON TEMPORARY;


/
GRANT CONNECT, RESOURCE TO
     comercios
/
```

En nuestro *script* de instrucciones ejecutaremos primero las instrucciones de borrado. De este modo si fuese necesario volver a lanzar el *script* no tendremos que preocuparnos por los mensajes de error del tipo "el objeto ya existe". Una vez conectados con un usuario con derechos de administración (por defecto todas las bases de datos *Oracle* crean un usuario *system* con la contraseña *manager*) podremos crear nuestro nuevo usuario **comercios** con la contraseña **oracle**. En la creación de dicho usuario especificaremos cuál es su *tablespace* de datos y el temporal. De tal manera, cuando el usuario se conecte y ejecute una instrucción del tipo *CREATE TABLE*, dicha tabla se almacenará por defecto (en nuestro caso especificaremos el *tablespace* no haciendo uso de esta posibilidad) en un segmento que automáticamente se creará en el *tablespace* de datos. El listado específico con esta especificación aparece incluido en el *CD-ROM*.

SQL SERVER

SQL Server es el gestor de base datos estrella de *Microsoft*. Con la última versión de este producto (7) *Microsoft* ha dado otro paso hacia delante acercándose a la cabeza del mercado de gestores de bases de

datos, cabeza que hasta la fecha veía un tanto lejana.

Recapitulando del número anterior, diremos que la arquitectura física de una base de datos está compuesta por el conjunto de ficheros y estructuras de datos que la componen y la arquitectura lógica por los objetos que los usuarios manejan.

Una base de datos *SQL Server* está físicamente implementada por dos o más ficheros de disco. Cada instalación de *SQL Server* tiene múltiples bases de datos: cuatro de sistema y al menos una de usuario. *SQL Server* diferencia entre bases de datos de sistema y de usuario; las de usuario serán las que nosotros crearemos y donde se almacenarán nuestros datos.

BASES DE DATOS DE SISTEMA

Una base de datos de sistema es la que *SQL Server* utiliza para propósitos internos, siendo otra de las grandes diferencias con respecto a *Oracle*. En el primero las bases de datos de sistema son compartidas por el resto de bases de datos de usuario, mientras que en *Oracle* al no ser así, la independencia de una base de datos con otra es total. Las cuatro bases de datos de sistema de *SQL Server* son:

- *master*
- *tempdb*
- *model*
- *msdb*

MASTER

Base de datos que almacena todos los niveles de información de un sistema *SQL Server*. Todas las cuentas de *login* y la configuración del sistema. Es la base de datos donde se almacena la existencia de las otras bases de datos (llamadas de usuario) y la localización de los ficheros de datos primarios que contienen la información de inicialización de cada una de ellas. Este modelo resulta extremadamente importante, por lo tanto es necesario disponer siempre de una copia de seguridad.

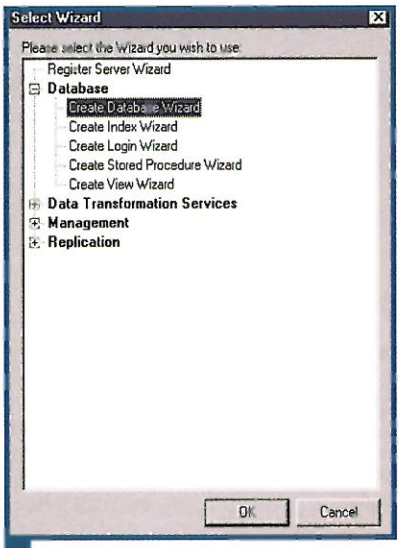


Figura 1.- Asistente para la creación de bases de datos SQL Server.

TEMPDB

Esta es la base de datos en la cual se almacenan todos los datos temporales, no importa qué usuario o base de datos las cree.

MODEL

Esta base datos contiene un modelo que se utilizará en la creación del resto de las bases de datos de usuario. Cuando se realice una instrucción *CREATE DATABASE* el primer paso de creación de esta base datos se realizará copiando el contenido de la base de datos modelo. Contiene la estructura de las tablas de sistema que son necesarias para cada base de datos. Su modificación permite personalizar las futuras bases de datos que se creen.

MSDB

Esta base de datos es utilizada por *SQL Server* en la planificación de tareas de alerta y de *jobs* (tareas programadas en el tiempo), siendo utilizada por el agente de *SQL Server*.

Figura 2.- Ventana con los pasos que se han de seguir con el asistente para la creación de bases de datos *SQL Server*.

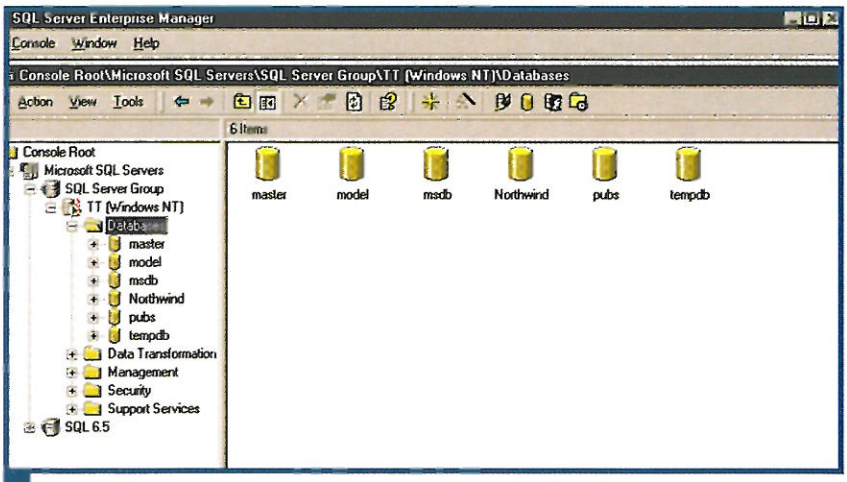


Figura 3.- Bases de datos creadas por defecto en la instalación de SQL Server, donde aparecen las bases de datos de sistema.

ARQUITECTURA FÍSICA

Todos los datos manejados por *SQL Server* se almacenan utilizando básicamente dos estructuras de datos: páginas y extensiones.

PÁGINAS

El tamaño de una página es de **8** *Kb*. En ellas se almacenan tanto los datos de índices como la información del catálogo. Cada página está referenciada mediante un identificador único. Contienen una cabecera de **96** *bytes* con información sobre el espacio libre en cada página, el identificador del objeto al que la página pertenece, ubicación física, el tipo de página y punteros a las páginas anteriores y posteriores.

EXTENSIONES

Una extensión es un conjunto de 8 páginas contiguas. La extensión es la unidad mínima de almacenamiento en las bases de datos. Cuando creemos una tabla o un índice, se le asociará una extensión. Cuando el tamaño de dicha tabla alcance **64** *Kb* (8 páginas) se le asignará lo que se denomina "extensión uniforme". Es decir, una extensión completa asignada únicamente a dicha tabla. Hasta ese momento habrá estado almacenada en una "extensión mixta", compartiendo dicha extensión con otras estructuras de datos.

TIPOS DE PÁGINAS

Clasificándolas por su contenido existen diversos tipos de páginas:

- Páginas de datos
- Páginas de texto de imagen
- Páginas de índice
- *Global Allocation Map Pages*
- Páginas de espacio en páginas
- *Index Allocation Map pages*

PÁGINAS DE DATOS

Los registros de nuestras tablas y el resto de los objetos de la base datos se almacenan en estas páginas. Un registro no puede ser almacenado en distintas páginas, por lo tanto, nos encontramos con una limitación en el tamaño de los registros. Con esta limitación perderíamos toda la potencia que los tipos de datos de gran tamaño *(text,ntext* e imagen) nos permiten. Es por este motivo que dichos tipos de datos son almacenados en páginas separadas.

PÁGINAS DE TEXTO DE IMAGEN

El tipo de *datos text* permite almacenar cadenas que contengan más de ocho mil caracteres (para cadenas de menor tamaño podemos utilizar el tipo de datos *varchar*), hasta un límite de 2.147.483.647 caracteres. Por su parte, el tipo de datos *Image* es utilizado para el almacenamiento de grandes bloques binarios, hasta un límite de 2.147.483.647 *bytes*.

PÁGINAS DE ÍNDICE

Almacena las tablas de punteros a los registros (índices).

GLOBAL ALLOCATION MAP PAGES

Contienen información acerca de las extensiones que están ocupadas y de las extensiones libres. De hecho, la información de la distribución de las extensiones uniformes (*uniform extent*) y de extensiones mixtas (*mixted extent*) están separadas en dos mapas diferentes dentro de estas mismas páginas.

PÁGINAS DE ESPACIO EN PÁGINAS

En ellas se almacena la información necesaria para conocer qué páginas están en estado libre (sin asignar) y qué páginas de datos han sido asignadas a los objetos. En el caso de una extensión ocupada parcialmente, la información almacenada en las *Global Allocation Map Pages* no es suficiente para conocer qué páginas están disponibles y cuál es el grado de ocupación en esta extensión. En este caso se hace necesario que el sistema recurra a este tipo de páginas para obtener dicha información.

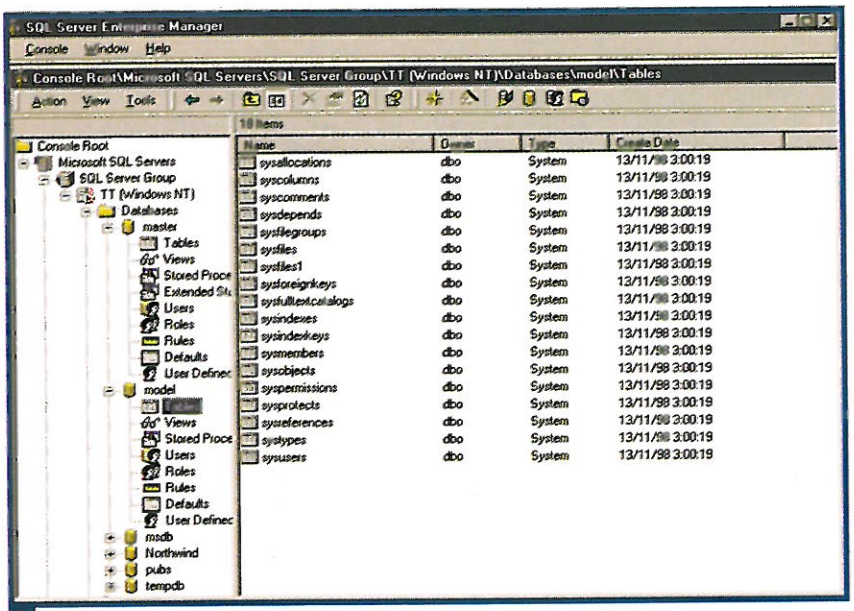


Figura 4.- Tablas de la base de datos Model.

INDEX ALLOCATION MAP PAGES

En ellas se almacena la información que asocia los objetos de nuestra base de datos a cada página, e información de las extensiones usadas por una tabla o índice. Cada una de estas páginas contiene la asignación de un rango de extensiones para un objeto en el fichero de datos.

FICHEROS DE DATOS

En la versión *6.5* los datos se almacenaban en dispositivos de datos, en la versión 7 han sido sustituidos por ficheros del sistema operativo. En un dispositivo de *SQL Server 6.5* podríamos almacenar diversas bases de datos o distribuirlas en diferentes dispositivos. En la nueva versión de *SQL Server*, al igual que en *Oracle*, almacenamos independientemente cada base de datos en un conjunto de ficheros (en lo que a los datos se refiere, ya que a nivel de sistema ya hemos visto que nuestras bases de datos son dependientes de las de sistema de *SQL Server*). De este modo, un archivo de datos no puede pertenecer simultáneamente a dos bases de datos.

La arquitectura física de *SQL Server* incluye un número de ficheros en disco entre los que podemos destacar tres tipos:

- Fichero de datos primario: Es el punto de comienzo de una base de datos. Todas las bases de datos tienen un único fichero de este tipo. En él se almacenan las tablas del sistema.

- Ficheros de datos secundarios: abarca a todos los ficheros de datos que no son de tipo primario. Son opcionales, permiten ubicar objetos en lugares específicos, así como el crecimiento de la base de datos a nuevos dispositivos físicos (como un nuevo disco duro).

- Ficheros de *log:* En ellos se guarda la información de *log* necesaria para recuperar la base de datos. Debe existir al menos un fichero de *log* por cada base de datos, aunque puede haber más de uno.

Podremos decir que existen dos nombres para cada fichero de una base de datos. El nombre lógico y el nombre físico. El primero será el utilizado para referirse a dicho fichero en todas las sentencias *SQL Server*. El segundo será el que identificará dicho fichero desde el sistema operativo. *SQL Server* no fuerza las extensiones en sus distintos tipos de fichero, pero propone una nomenclatura para identificarlos:

- *mdf* para los ficheros de datos primarios
- *ndf* para los ficheros de datos secundarios
- *ldf* para los ficheros de *log*

GRUPOS DE FICHEROS

Los ficheros pueden agruparse en grupos, con el fin de facilitar la administración y algunas tareas como puede ser la de *backup*. La estrategia de *SQL Server* a la hora de almacenar datos está basada en los grupos de ficheros, no en ficheros individuales. La distribución de los datos en cada fichero perteneciente al grupo es equilibrada. es decir, cuando sea necesario reservar espacio para el almacenamiento de un nuevo objeto en un grupo de ficheros se tendrá en cuenta el espacio libre en cada fichero.

El rendimiento de base datos también puede apoyarse en esta característica. Controlando el almacenamiento de los datos en un disco y los índices en otro logramos paralelizar la entrada y salida a disco. El administrador puede crear grupos de ficheros para cada disco duro y

Tabla 1. Equivalencia entre la terminología Oracle y SQL Server.

Oracle	SQL Server
archived redo log	backup transaction log
control file	master database
data block	Page
data dictionary	database catalog
data dictionary view	system table
export file	data file
library cache	procedure cache
online redo log	transaction log
redo buffer	log buffer
redo log file	backup file
sequence	identity column
system global area (SGA)	memory buffers
tablespace	Filegroup

asignar cada uno de ellos a una tabla o índice en concreto.

CREACIÓN DE UNA BASE DE DATOS SQL SERVER

Antes de comenzar el proceso de creación hemos de considerar algunos aspectos iniciales: el nombre de la base de datos, su propietario, su tamaño y los ficheros donde se almacenará. Siempre es recomendable realizar una copia de seguridad de la base de datos **master** antes y después de la creación de una base de datos de usuario.

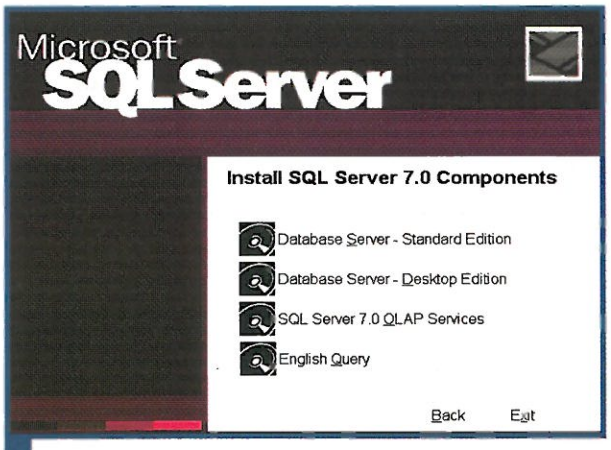


Figura 5.- Instalación del SQL Server.

A la hora de crear nuestra base de datos podemos utilizar diversos métodos. Bien utilizando *Transact-SQL* (sentencias *SQL Server*), el *SQL Server Enterprise Manager* (herramienta gráfica de administración de *SQL Server*), o bien un asistente que nos guiará durante todo el proceso de creación.

ON

Tras esta cláusula se especifican cuáles son los ficheros de datos y el fichero primario. Por cada uno de estos ficheros se especifica el nombre lógico (cláusula *NAME*), nombre físico del fichero (cláusula *FILENAME*) y el tamaño inicial del fichero (cláusula *SIZE*). También podemos configurar nuestra base de datos para que sus ficheros tengan un crecimiento automático, para ello indicaremos un valor máximo tras la clausura *MAXZIZE*; con *FILEGROWTH* indicaremos el tamaño del incremento.

TRANSACT-SQL

En los gestores relacionales el lenguaje más usado es la estandarización *ANSI* de *SQL*, *SQL Server 7* utiliza una variante denominada *Transact-SQL*. Haciendo uso de este lenguaje la sentencia para creación de una base datos es la siguiente:

```
CREATE DATABASE database_name
    [ ON [PRIMARY]
    [ <lista de ficheros> [,...n]
    ]
    [, <lista de grupos> [,...n]
    ]
    ]
    [ LOG ON { <lista de fiche-
    ros> [,...n]} ]
    [ FOR LOAD | FOR ATTACH ]


< lista de ficheros> y <lista de
    grupos> los sustituiremos
    por:

<lista de ficheros> ::=

( [ NAME = nombre lógico, ]
    FILENAME = 'nombre físico'
    [, SIZE = tamaño]
    [, MAXSIZE = { tamaño máximo
    | UNLIMITED } ]
    [, FILEGROWTH = incremento] )
    [,...n]

<lista de grupos> ::=

FILEGROUP nombre del grupo <lista
    de ficheros> [,...n]

database_name
```

Es el nombre de la nueva base de datos, que debe ser menor de **728** caracteres.

LOG ON

Tras esta cláusula se especifican qué ficheros serán utilizados para almacenar el *log* de la base de datos. En caso de no específicarlo se creará un único fichero de *log* automáticamente, con un nombre generado por el sistema y un tamaño igual al **25%** de la suma de todos los ficheros de datos.

SUSCRÍBETE

A SÓLO PROGRAMADORES

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

- NOTICIAS
- MULTIMEDIA
- TEORÍA
- INTERNET
- DESARROLLO DE APLICACIONES

QUE NO TE LÍEN CON EL <CÓDIGO>

- COMUNICACIONES
- HERRAMIENTAS
- PROGRAMACIÓN
- ÚLTIMAS TENDENCIAS
- Y MUCHO MÁS...

BOLETÍN DE SUSCRIPCIÓN

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L. (Revista Sólo Programadores).
C/ San Sotero, 5. 1ª Planta. 28037 Madrid. Tlf: 91 304 87 64. Fax: 91 327 13 03

Quiero suscribirme a la revista SÓLO PROGRAMADORES desde el Nº.....
y beneficiarme de las condiciones de estas magníficas promociones:

❑ **SUSCRIPCIÓN ANUAL**
12 NÚMEROS + 12 CD-ROMs
AL PRECIO DE
13.720 ptas. / 82,45 €

❑ **SUSCRIPCIÓN ANUAL ESPECIAL ESTUDIANTES**
12 NÚMEROS + 12 CD-ROMs
por sólo 10.876 ptas. / 65,36 €

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❑ Domiciliación bancaria
❑ Contra reembolso

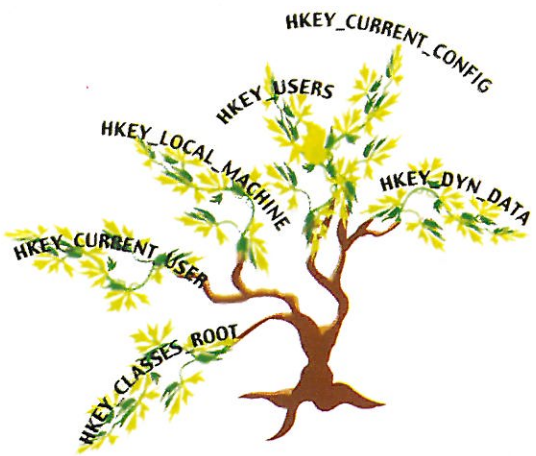
Soy antiguo suscriptor
❑ **Sí** ❑ **No**

PARA ENVÍOS AL EXTRANJERO SÓLO SE ADMITIRÁN LAS SIGUIENTES FORMAS DE PAGO:
❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❑ Eurocheque conformado con un banco español a nombre de REVISTAS PROFESIONALES, S.L

NOMBRE Y APELLIDOS:.....
EDAD:..... PROFESIÓN: .....
TFNO: ..... DOMICILIO: .....
CIUDAD:..... C.P: .....PROVINCIA:.....

Datos de domiciliación:
Banco.....
Domicilio .....
Nº de Cuenta.....
Titular.....
Fecha .....
FIRMA

Promoción válida hasta agotar existencias



Programación del registro de Windows (y III)

Adolfo Aladro (aaladro@arrakis.es)

En este último artículo vamos a tratar todas aquellas funciones que permiten escribir en el Registro de *Windows*. Con esta entrega estaremos en disposición de hacer que nuestras propias aplicaciones tengan una funcionamiento más profesional, utilizando el registro de *Windows* como una zona de almacenamiento permanente de diversos datos de configuración.

"Hombre prevenido vale por dos"

A lo largo de esta serie hemos puesto especial hincapié en el hecho de que el Registro de *Windows* es algo bastante delicado. Cualquier equivocación a la hora de escribir en él puede ocasionar trastornos muy graves. Hasta el momento se ha aprendido a leer información del Registro. Si la aplicación encargada de hacer esto comete un error, en principio no hay ningún motivo para que ocurra nada desastroso. En cambio, si escribimos, borramos o modificamos alguna entrada del Registro el coste de un error puede ser muy caro. Por eso es de vital importancia hacer una copia de seguridad antes de comenzar a programar una aplicación para poner en práctica estos conocimientos.

Lo primero es realizar una copia de seguridad del Registro

En el primer artículo vimos dos formas sencillas de crear una copia de seguridad del Registro. Antes de seguir adelante, sería conveniente ponerlas en práctica. Ya reza el refrán: "Hombre prevenido vale por dos".

Creación de una nueva clave (RegCreateKeyEx)

La función que vamos a ver ahora, *RegCreateKeyEx,* sirve para crear una clave, en el caso de que no exista, o para abrirla en caso de que ya exista.

```
Declare Function RegCreateKeyEx
    Lib
  "advapi32.dll" Alias
   "RegCreateKeyExA"
  (ByVal hKey As Long,
   ByVal lpSubKey As String,
   ByVal lpReserved As Long,
   ByVal lpClass As String,
   ByVal dwOptions As Long,
   ByVal samDesired As Long,
   lpSecurityAttributes As
       SECURITY_ATTRIBUTES,
   PhkResult As Long,
   lpdwDisposition As Long) As
   Long
```

- *hKey:* Identificador de la clave raíz a partir de la cual se va a crear la nueva rama. Será una de las constantes que identifican las ramas del Registro: *HKEY_CLASSES_ROOT*, *HKEY_CURRENT_CONFIG*, *HKEY_CURRENT_USER*, *HKEY_DYN_DATA*, *HKEY_LOCAL_MACHINE* o *HKEY_USERS*.

lpdwDisposition determina si se ha abierto una clave existente o si se ha creado

- *lpSubKey:* Nombre de la clave que se va a crear, o abrir en el caso de que ya exista. El formato ha de ser similar al que se utiliza en la función *RegOpenKeyEx*.
- *lpReserved:* Parámetro reservado (Debe dejarse siempre con valor igual a **0**).
- *lpClass:* Nombre de clase o descripción de la clave.
- *dwOptions:* Parámetro relativo a la creación de la clave. Existen dos posibles valores: *REG_OPTION_VOLATILE* y *REG_OPTION_NON_VOLATILE*. El primero de los dos implica que la clave que se va a crear no se escribirá de manera definitiva en el Registro. Por el contrario, si se escoge la opción *REG_OPTION_NON_ VOLATILE* el Registro guardará esa clave de forma permanente. Solamente *Windows NT* permite la creación de claves volátiles. Estas claves desaparecen cuando el sistema se reinicia.

Si el sistema es Windows NT se pueden establecer permisos con respecto a las claves

- *samDesired:* Identificador de los permisos de la clave que se va a abrir o crear. Los valores que puede adoptar son los descritos para la función *RegOpenKeyEx*.
- *LpSecurityAttributes:* Sólo tiene sentido cuando estamos trabajando con *Windows NT* y en él se indican atributos de seguridad relacionados con la clave. Si estamos trabajando con *Windows 95/98* se ignorará el valor de este parámetro.
- *PhkResult:* Identificador de la clave abierta o creada.
- *LpdwDisposition:* El valor de la variable indicada como último parámetro servirá para saber si se ha abierto una clave existente o bien si se ha creado una nueva: *REG_OPENED_EXISTING_KEY* o *REG_CREATED_NEW_KEY*.

Al igual que ocurría con la mayoría de las funciones que hemos visto hasta el momento, *RegCreateKeyEx* devuelve **ERROR_SUCCESS** en el caso de que la operación haya concluido con éxito, o cualquier otro valor si ha habido algún error.

REVISIANDO NUESTRO PRIMER EJEMPLO

El primer ejemplo que apareció en esta serie consistía en un pequeño programa que servía para abrir ramas del Registro. No es que fuera

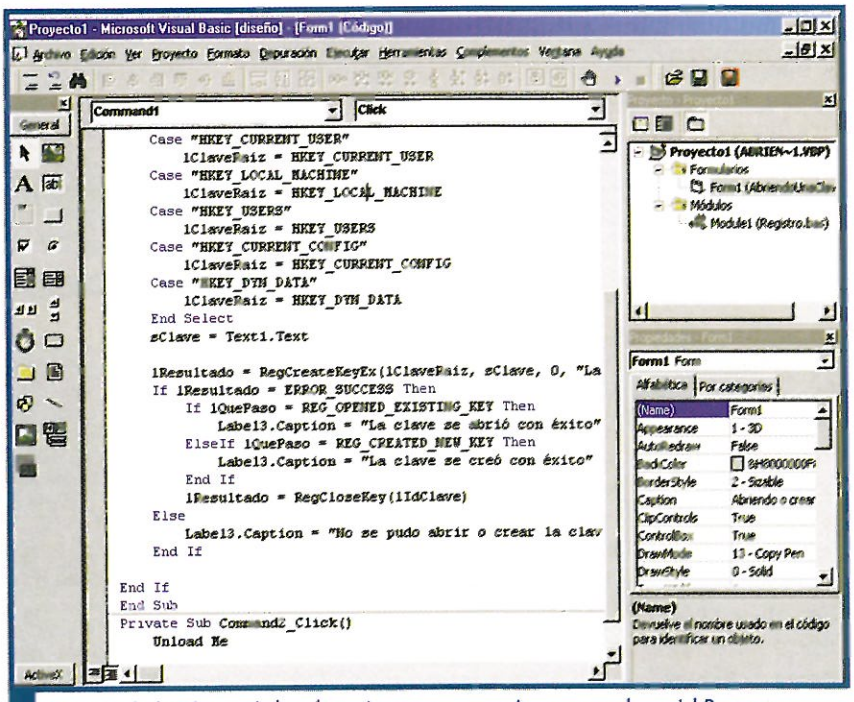


Figura 1.- Código fuente de la aplicación que sirve para abrir o crear claves del Registro.

muy complejo – ni muy útil - pero sirvió como introducción.

Ahora que hemos visto la función *RegCreateKeyEx* podemos reescribir el programa de manera que sirva para abrir una clave en el caso de que exista o bien para crearla directamente.

Casi todos las aplicaciones profesionales escriben en el Registro de Windows

La Figura 1 muestra el núcleo principal de nuestro pequeño programa. En vez de llamar a *RegOpenKeyEx* se utiliza la función que acabamos de ver. De esta manera pueden ocurrir dos cosas: si la clave ya existe, simplemente se abre; en caso contrario se crea una clave nueva. Lo más importante es reparar en la llamada a la función *RegCreateKeyEx*.

```
LResultado =
     RegCreateKeyEx(lClaveRaiz,
     sClave, 0, "Lala",
     REG_OPTION_NON_VOLATILE,
     KEY_ALL_ACCESS, 0, lIdClave,
     lQuePaso)
```

Las Figuras 2 y 3 ilustran los diferentes resultados que proporciona nuestra aplicación de ejemplo dependiendo de la clave que se introduce como parámetro. Finalmente la Figura 4 contiene un extracto del Registro, visualizado con la herramienta *regedit*, que refleja la nueva clave creada desde la aplicación.

La clave de estos ejemplos no ha sido elegida al azar. Como ya sabemos, existe una rama del registro de *Windows* donde se guarda toda la información de desinstalación de los programas.

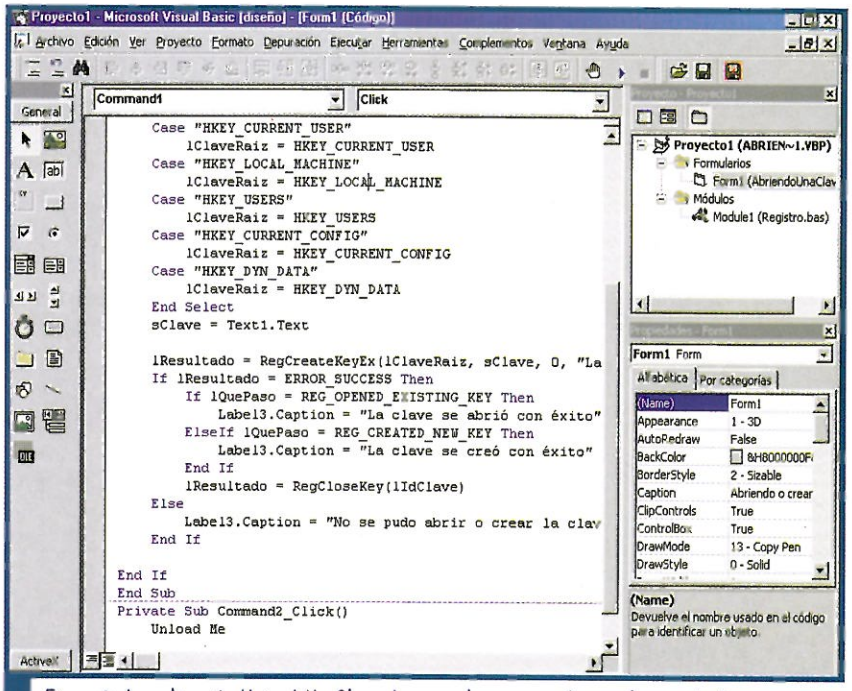
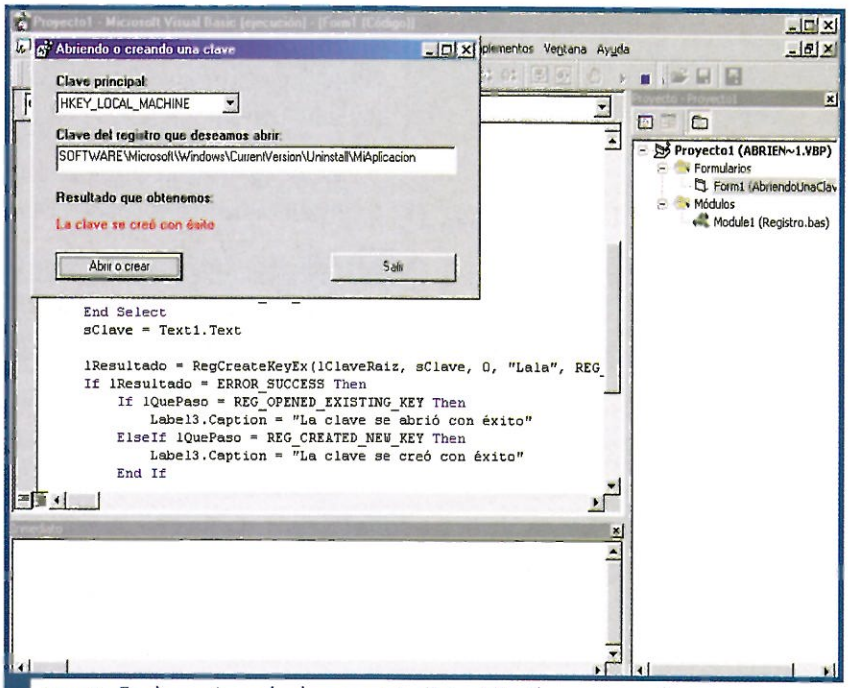


Figura 2.- La aplicación AbriendoUnaClave abre una clave, en vez de crearla, cuando ésta ya existe.



igura 3.- En el caso de que la clave no exista, AbriendoUnaClave crea una clave.

```
HKEY_LOCAL_MACHINE\SOFTWARE\
     Microsoft\Windows\CurrentVersion\
     Uninstall\ EasyCleaner
```

Si queremos realizar una aplicación nos ofrezca la desinstalación automática, tendremos que poder escribir en esta rama del Registro, como hacen las aplicaciones profesionales. Ya hemos dado el primer paso que consiste en crear una entrada en la rama del Registro.

ESCRITURA DE UN VALOR (REGSETVALUEEX)

Escribir un valor en el registro se hace de manera similar a como se lee. En este caso el procedimiento de la *API* de *Windows* que realiza esta tarea se denomina *RegSetValueEx*.

```
Declare Function RegSetValueEx Lib
   "advapi32.dll" Alias
     "RegSetValueExA"
   (ByVal hKey As Long,
    ByVal lpValueName As String,
    ByVal lpReserved As Long,
    ByVal dwType As Long,
    lpData As Any,
    ByVal cbData As Long) As Long
```

- *hKey:* Identificador de la clave que contiene el valor que deseamos escribir.
- *lpValueName:* Nombre del valor dentro de la clave.
- *lpReserved :* Parámetro reservado (Debe dejarse siempre con valor igual a **0**).
- *lpType:* Variable donde se almacena el tipo de la variable.
- *lpData:* Cadena donde se guarda el contenido del valor.
- *lpcbData:* Número que expresa el tamaño de la cadena anterior.

Es importante observar que el contenido del valor se pasa por valor (ByVal)

RegSetValueEx toma exactamente los mismos parámetros que la función *RegQueryValueEx*. La diferencia estriba en que ahora somos nosotros los que tenemos que proporcionar la información que debe escribirse en el Registro. Si el valor que se pretende escribir no existe en la clave se creará una nueva.

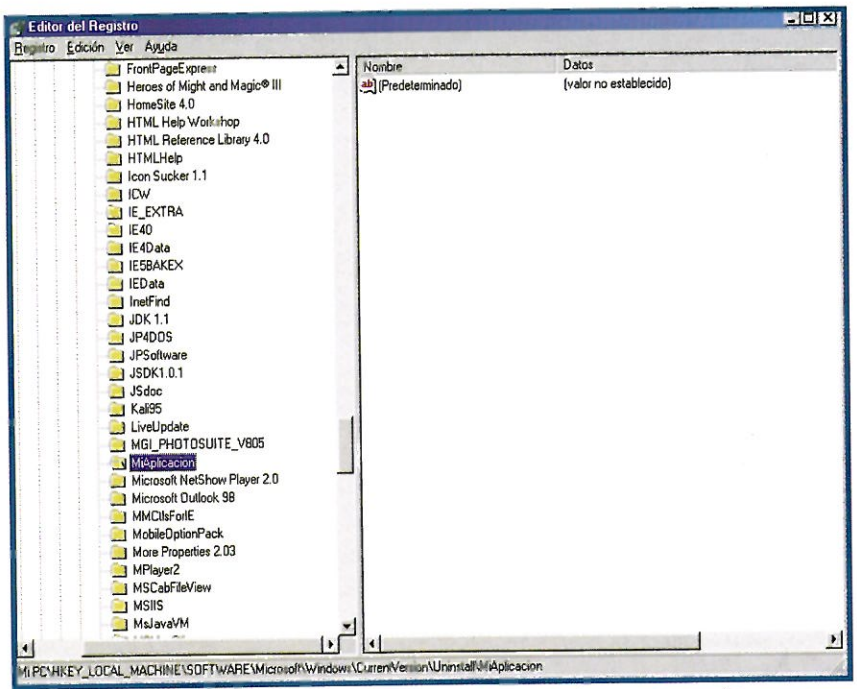


Figura 4.- Resultado de escribir en el Registro desde nuestra aplicación en *Visual Basic*.

Siguiendo con el ejemplo del apartado anterior vamos a ver cómo es posible insertar la información necesaria en el registro de *Windows* para que la opción de desinstalación automática aparezca en la opción *Agregar o quitar programas* del *Panel de Control* de *Windows*.

Además de la entrada correspondiente a la aplicación que cuelga de la rama arriba indicada, deben existir un par de valores en dicha entrada: *DisplayName* y *Uninstall String*. El primero de ellos contiene una cadena de texto con la descripción del programa que aparece en la pestaña de desinstalación de *Windows*. El segundo representa una ruta absoluta al ejecutable que se encarga de desinstalar la aplicación.

La nueva versión del programa de nuestro ejemplo permite crear la clave en el Registro e insertar los valores adecuados. Una vez que hemos hecho esto, podemos comprobar inmediatamente, tal y como ilustra la Figura 5, que en la pestaña *Instalar o desinstalar* de *Windows* aparece una referencia al programa deseado. Asimismo, la Figura 6 muestra el contenido de la nueva entrada del Registro, con los valores *DisplayName* y con *UninstallString*.

ASOCIAR ARCHIVOS

Cuando se instalan aplicaciones en nuestro ordenador normalmente éstas registran algunas extensiones de archivos. Así cuando hacemos doble *clic* en el explorador de *Windows* sobre un fichero con alguna de esas extensiones registradas se abre de forma automática la aplicación. Esto se logra estableciendo una serie de parámetros en el Registro de *Windows*.

Dentro de la clave raíz *HKEY_CLASSES_ROOT* se encuentran una serie de subclaves del

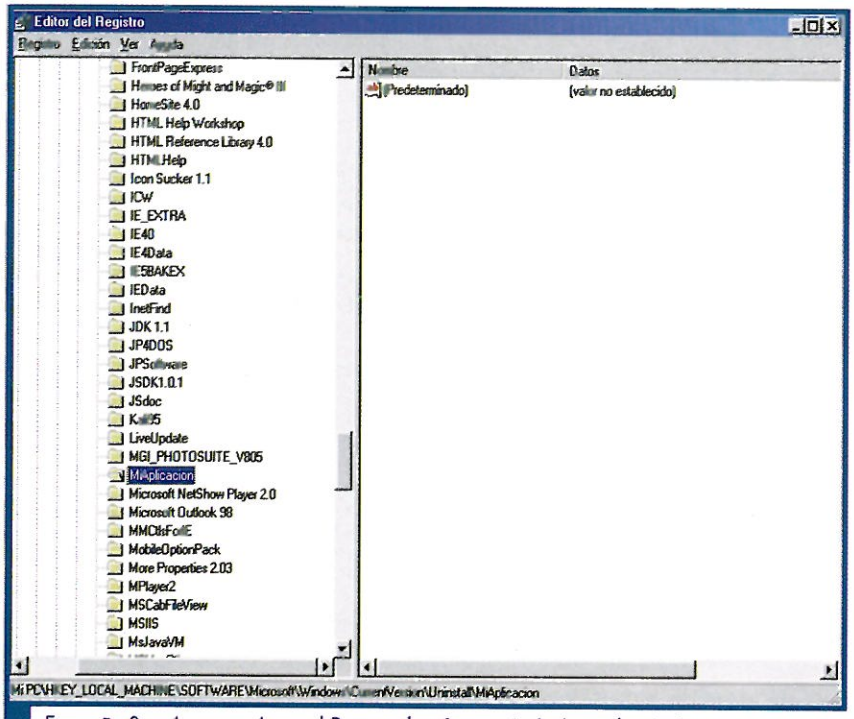


Figura 5.- Cuando se escribe en el Registro la información de desintalación de un programa, éste aparece en la pestaña Instalar o desinstalar de *Windows*.

tipo *.txt*. Esto hace que el sistema reconozca los archivos con extensión *txt*. Entre otra muchas cosas, a través de este sistema de configuración es posible asociar aplicaciones a las extensiones. Así el valor predeterminado que se encuentra en la siguiente clave:

```
HKEY_CLASSES_ROOT\.txt\shell\
    open\command
```

podría ser:

```
C:\WINDOWS\NOTEPAD.EXE %1
```

Nota: Nótese que en el caso de que deseemos escribir el valor predeterminado, se accede a él del mismo modo que hacía para leerlo: simplemente asignando una cadena vacía a la llamada RegSetValueEx.

Si desarrollamos una aplicación que utilice sus propios tipos de archivos lo más elegante es sin duda alguna modificar el Registro de *Windows* durante la instalación, o la primera vez que se ejecute, para asociar las extensiones. Para ellos se utilizarán las llamadas *RegCreateKeyEx* y *RegSetValueEx*.

Para asociar archivos es preciso acceder al Registro

Otra de las características que puede ser interesante es que las nuevas extensiones registradas aparezcan en el menú contextual *Nuevo* que aparece cuando se hace *clic* con el botón derecho del ratón. Para ello será necesario agregar una nueva subclave:

```
HKEY_CLASSES_ROOT\.txt\ ShellNew
```

Dentro de ella deberá aparecer un valor cuyo nombre será *NullFile* y su valor será "", o sea, una cadena vacía.

Nos quedan por ver dos llamadas de la API de Windows que permiten borrar un valor y una clave.

BORRADO DE UN VALOR (REGDELETEVALUE)

La función *RegDeleteValue* permite borrar un determinado valor.

```
Declare Function RegDeleteValue Lib
   "advapi32.dll" Alias
     "RegDeleteValueA"
   (ByVal hKey As Long,
    ByVal lpValueName As String) As
     Long
```

- *hKey:* Identificador de la clave que contiene el valor que deseamos escribir.
- *lpValueName:* Nombre del valor dentro de la clave.

BORRADO DE UNA CLAVE (REGDELETEKEY)

La función *RegDeleteKey* permite borrar un determinado valor.

```
Declare Function RegDeleteKey Lib
   "advapi32.dll" Alias
     "RegDeleteKeyA"
   (ByVal hKey As Long,
    ByVal lpSubKey As String) As Long
```

- *hKey:* Identificador de la clave raíz en la que se encuentra la rama a eliminar. Como el resto de las llamadas de la *API* de *Windows* será una de estas: *HKEY_CLASSES_ROOT*, *HKEY_CURRENT_CONFI*, *HKEY_CURRENT_USER*, *HKEY_DYN_DATA*, *HKEY_LOCAL_MACHINE* o *HKEY_USERS*.
- *LpSubKey:* Nombre de la clave que se va a eliminar.

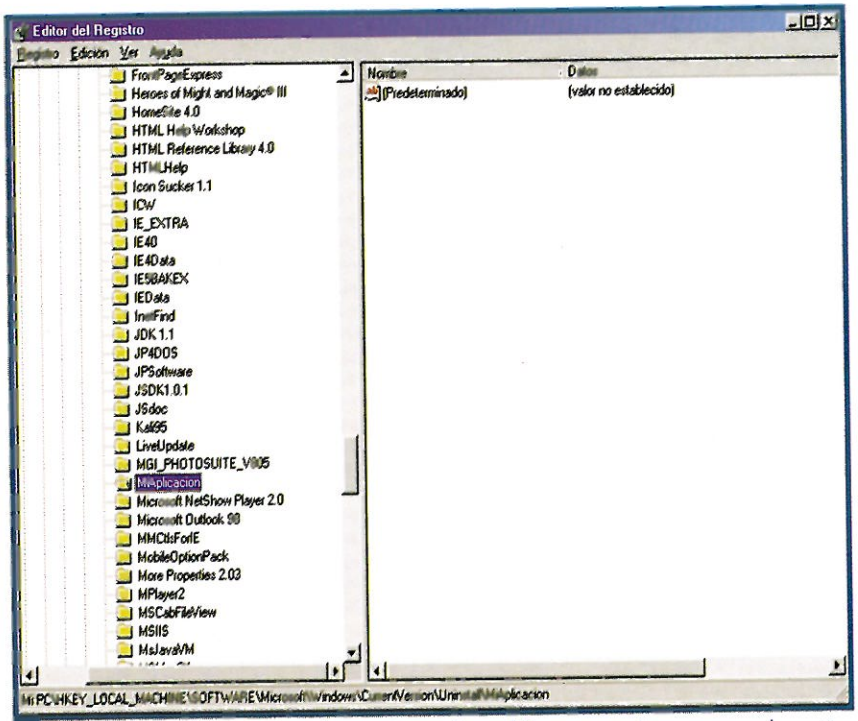


Figura 6.- Aspecto que presenta la entrada creada en el Registro para hacer aparecer la opción de desinstalación de un programa.

Listado 1. Código que permite crear un valor en una determinada clave del Registro de Windows.

```
If Combo2.ListIndex <> -1 Then

    Select Case Combo2.List(Combo2.ListIndex)
    Case "REG_BINARY"
        lTipo = REG_BINARY
    Case "REG_DWORD"
        lTipo = REG_DWORD
    Case "REG_SZ"
        lTipo = REG_SZ
    Case "REG_MULTI_SZ"
        lTipo = REG_MULTI_SZ
    Case "REG_NONE"
        lTipo = REG_NONE
    End Select

    lResultado = RegOpenKeyEx(lClaveRaiz,
        sClave, 0, KEY_ALL_ACCESS, lIdClave)
    If lResultado = ERROR_SUCCESS Then
        Label3.Caption = "La clave se abrió con éxito"

        sNombreValor = Text2.Text
        sValor = Text3.Text
        lResultado = RegSetValueEx(lIdClave,
            sNombreValor, 0, lTipo,
            ByVal sValor, Len(sValor))
        If lResultado = ERROR_SUCCESS Then
          Label8.Caption = "El valor se escribió con éxito"
        Else
          Label8.Caption = "Error al crear el valor"
        End If
        Lresultado = RegCloseKey(lIdClave)
    Else
        Label3.Caption = "No se pudo abrir la clave"
    End If
End If
```

PASO DE PARÁMETROS

Por último, es importante prestar atención a la declaración de las funciones que proporciona el *Visor de texto API*, especialmente para observar de qué manera pasan cada uno de los parámetros. Si no nos ajustamos a él la función probablemente fallará.

En *Visual Basic*, y en general en casi todos los lenguajes de programación, existen dos maneras de pasar los parámetros a una función:

- Por valor (*ByVal*): Se pasa a la función el valor del argumento en vez de su dirección.
- Por referencia (*ByRef*): Se pasa la dirección en vez de su valor. Modo predeterminado en *VB*.

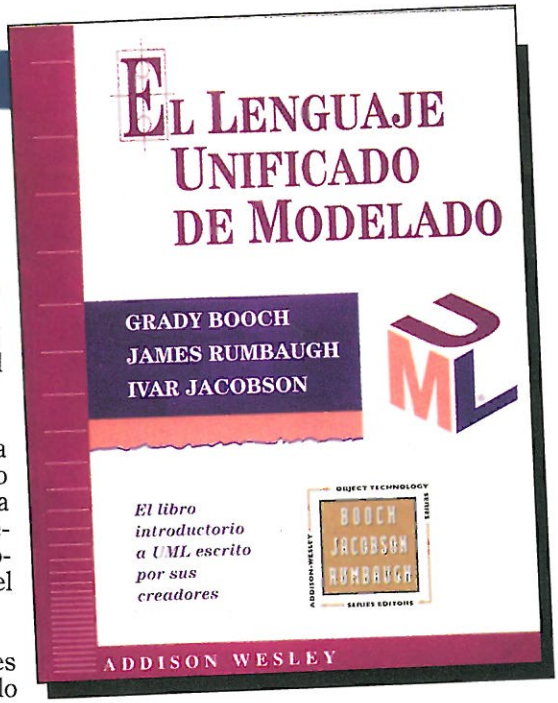
La función RegDeleteKey permite borrar una clave y todo su contenido

Son muchas las implicaciones que tiene la forma en la que se pasan los parámetros, pero su estudio va mucho más allá de esta serie de artículos. Lo fundamental, de cara a utilizar las llamadas de la *API* de *Windows*, para acceder al Registro de *Windows*, es que se respeten las declaraciones de las funciones. Así por ejemplo en el código fuente de nuestro ejemplo hacemos:

```
lResultado = RegSetValueEx(lIdClave,
    sNombreValor, 0, lTipo, ByVal
    sValor, Len(sValor))
```

Anteponiendo la palabra reservada *ByVal* delante de la variable que contiene el contenido del valor se indica, explícitamente, que se desea que esa variable se pase por valor en vez de por referencia, que es tal y como se haría por defecto.

EL LENGUAJE UNIFICADO DE MODELADO



El *UML* se ha convertido en el lenguaje gráfico estándar para especificar, construir, visualizar y documentar sistemas con gran cantidad de *software*.

En este manual, los creadores originales de *UML* proporcionan una introducción a los elementos fundamentales del lenguaje en un formato especialmente diseñado para facilitar la lectura. Aplicándolo de manera progresiva a una serie de problemas de modelado cada vez más complejos a través de una amplia variedad de dominios de aplicación. Este enfoque guiado por ejemplos, ayuda a los lectores a comprender y aplicar el *UML* rápidamente.

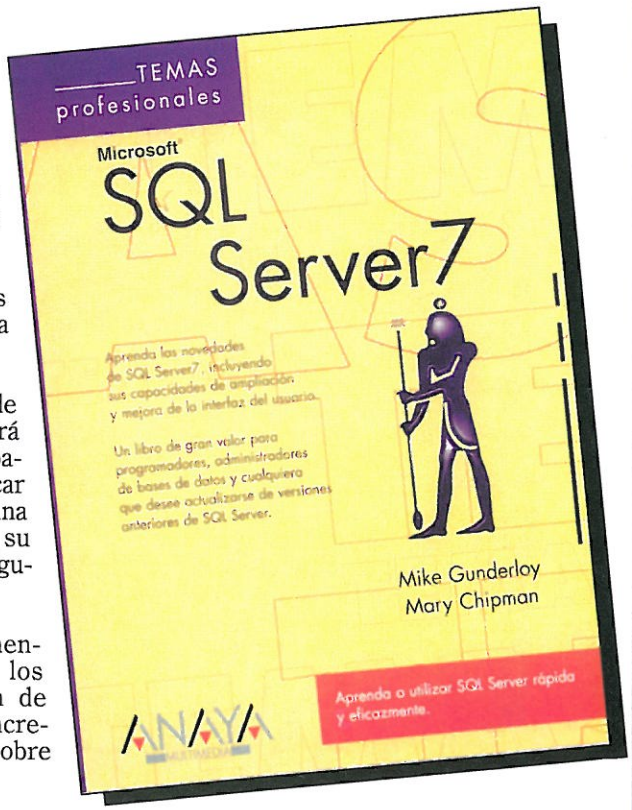
Para los desarrolladores más avanzados el libro incluye consideraciones acerca de cómo aplicar *UML* y por qué es importante para el desarrollo de sistemas con gran cantidad de *software*. Así conseguirán dominar el vocabulario, las reglas y las construcciones particulares de este lenguaje para conseguir su máxima efectividad.

Los usuarios aprenderán a aplicar el *UML* a una amplia variedad de problemas de modelado comunes gracias a los ejemplos preparados, adquiriendo una perspectiva obtenida de los diseñadores originales de este lenguaje.

Editorial: Prentice Hall/Addison Wesley
Nº páginas: 464
Nivel: Inicial-intermedio

Autor: Crady Booch, James Rumbaugh e Ivar Jacobson
Idioma: Español
Precio: 3.990 (I.V.A inc.)

SQL SERVER 7



Este ejemplar le ofrece una cobertura clara y práctica de la base de datos más importante de *Microsoft*.

Gunderlay y *Chipman*, autores del libro, son dos expertos que le enseñarán a diseñar, construir y manipular el *SQL Server* de bases de datos y sus aplicaciones.

Las instrucciones que aquí encontrará, acompañadas de los mejores tutoriales, le enseñarán a manejar el *SQL Server* rápidamente de una forma clara y sencilla.

Cada capítulo se centra en un caso real por lo que el lector puede seguir el orden de su lectura según su propio criterio e interés. Podrá aprender a evaluar un *SQL Server*, administrar bases de datos, trabajar con tablas, organizar, normalizar, mover, recuperar y modificar datos, utilizar las herramientas de *SQL Server*, editar los datos en una *Red* o recuperarlos; así como administrar un servidor, controlar su seguridad u optimizar la configuración de esta aplicación.

Editorial: Anaya
Nº de páginas: 576 aprox.
Nivel: Avanzado-Experto
Autor: Mike Gunderloy,
Mary Chipman
Idioma: Español
Precio: 4.995 (I.V.A inc.)

Este ejemplar está especialmente indicado para todos los usuarios de esta aplicación de bases de datos que deseen incrementar sus conocimientos sobre el tema.

VISUAL BASIC 6: PROGRAMACIÓN CLIENTE/SERVIDOR

Michael MacDonald y *Kurt Cagle*, los autores de este libro, nos muestran cómo dominar las características más importantes de *Visual Basic* para desarrollar aplicaciones cliente/servidor orientadas a la empresa.

Esta obra, de carácter avanzado, ha sido especialmente diseñada para guiar al lector de forma rápida por los conceptos fundamentales del desarrollo de redes y bases de datos y por la creación práctica de aplicaciones utilizando *OLE/DB, COM/DCOM* y el nuevo *ADO* (Objetos de Datos Activos), de *Microsoft*.

A lo largo del manual se hace especial hincapié en las técnicas de programación y diseño de bases de datos, además de enseñar a dominar todas las técnicas de desarrollo necesarias para crear proyectos orientados a objetos desde el escritorio, mediante *IIS* y aplicaciones basadas en la *Web*.

Con *Visual Basic 6* seremos capaces de desarrollar informes totalmente funcionales empleando el nuevo y potente objeto *DataReport*, diseñar y codificar aplicaciones *DHMTL* basadas en *Visual Basic* que puedan interactuar con bases de datos *ADO*, o crear procedimientos almacenados y disparadores para mejorar la potencia de nuestras aplicaciones. También tendremos la posibilidad de dominar las técnicas para poder utilizar controles de usuario vinculados a datos con el nuevo objeto *DataRepeater* entre otras muchas opciones.

Editorial: Paraninfo
Nº de páginas: 732
Nivel: Intermedio-Avanzado
Autor: Michael MacDonald, Kurt Cagle
Idioma: Español
Precio: 6300 (I.V.A. inc.)

REDES CON MICROSOFT TCP/IP

La editorial *Prentice Hall* nos ofrece una edición especial de Redes con *Microsoft*, con una referencia completa y actualizada para implementar *Microsoft TCP/IP* en su red.

El lector obtendrá ayuda experta para la instalación, la construcción de redes, la conexión a *Internet*, la integración de redes *Ethernet* y muchos otros temas.

Este ejemplar le ofrecerá información muy valiosa acerca de *TCP/IP* con la posibilidad de aprender a implementarlo en su organización de manera efectiva y sencilla.

La tercera edición de este manual le permitirá utilizar *Microsoft Proxy Server* para establecer accesos seguros a *Internet*; así como configurar *Microsoft Exchange Server* para gestionar *E-mail* o comprender con todo detalle las tecnologías *Microsoft* de servidores.

Además, el usuario será capaz de utilizar los diferentes servicios que proporciona *Internet Information Server 4.0*, optimizar el acceso a servidores remotos, instalar y configurar el *TCP/IP* o aprender a administrar *DHCP (Dynamic Host Configuration Protocol)*.

Editorial: Prentice Hall
Nº de páginas: 826
Nivel: Intermedio-Avanzado
Autor: Drew Heywood, MCSE
Idioma: Español
Precio: 5.500

Dudas técnicas

Cualquier duda o cuestión que queráis plantear tiene cabida en esta sección que confeccionáis vosotros mismos, los lectores de SÓLO PROGRAMADORES.

PREGUNTA

Estimados amigos:

Llevo ya varios años comprando vuestra revista, una de las mejores del mercado. Sin ninguna duda, la calidad de los artículos y la complejidad de algunos temas me han hecho abstenerme de realizar preguntas excesivamente básicas. Sin embargo, ha llegado el momento de romper el fuego. Espero que la duda que os envío sea lo suficientemente interesante como para que la publiquéis.

Mi pregunta es a la vez muy sencilla y bastante complicada. Llevamos ya más de un año desconcertados por la gran cantidad de microprocesadores que existen. En la época de los 486, uno podía distinguir su velocidad por los *Mhz*. Más *Mhz* equivalía a más velocidad.

Sin embargo, hoy en día existen diferentes modelos según el fabricante (*IBM, Cyrix, Amd* o *Intel*) y además, últimamente hay diferentes procesadores incluso dentro de la misma marca. ¿Son todos iguales por dentro?¿Hay diferencia entre las velocidades de unos y otros?

Finalmente, ¿Hay diferencias al programarlos? Teóricamente son todos compatibles *x86*, sin embargo he oído comentarios relativos a diferentes extensiones que hacen incompatibles entre sí a los diferentes *chips*. ¿Cómo influye a la hora de realizar programas?

Espero, como tantos otros lectores, vuestra respuesta.

RESPUESTA

Estimado lector:

Aunque tu pregunta se sale de lo que es estrictamente el campo de la programación, consideramos que sus implicaciones son tan relevantes que haríamos mal en no responderla, siempre desde nuestra orientación particular.

Vamos a intentar, de una manera muy genérica, comentar las diferencias tanto en cuestiones de velocidad como a la hora de programar las distintas *CPU's*.

● Velocidad, algo más que Mhz

Lo mejor para evaluar la velocidad de un microprocesador es observar el tiempo que tarda en ejecutar las aplicaciones que vamos a utilizar con mayor frecuencia. Si esta prueba es imposible nos queda la posibilidad de utilizar un banco de pruebas o *benchmark*.

De entre los muchos programas de este tipo disponibles, intentaríamos utilizar uno que se ajustara a nuestras necesidades. Existen *benchmarks* que evalúan el comportamiento global del sistema, el disco duro, el subsistema gráfico, etc... A la hora de ver qué *chip* es más rápido, deberemos elegir una aplicación que mida únicamente el microprocesador, ya que en caso contrario nos estaremos llamando a engaño con los resultados obtenidos.

Probablemente el *benchmark* de más conocido dedicado a evaluar únicamente la *CPU*, es el denominado *SPEC*. Podemos encontrar su descripción y resultados para casi todos los procesadores del mercado en la dirección *web*:

```
http://www.spec.org
```

En cualquier caso, ni el banco de pruebas más primitivo contempla un único dato para evaluar su velocidad. Normalmente se suelen utilizar dos magnitudes. La primera es lo que se denomina velocidad de Cálculo Entero (*Integer Performance*) y cuantifica la rapidez del procesador a la hora de ejecutar aplicaciones ofimáticas, bases de datos, etc... La segunda, la velocidad de Punto Flotante, que determina el rendimiento de la *CPU* a la hora de utilizar aplicaciones gráficas como *Photoshop* y similares, así como juegos y programas en *3D*.

Si observamos detenidamente los resultados obtenidos por *SPEC* podemos comprobar cómo procesa-

dores con una alta velocidad de reloj, son apenas más rápidos que otros que podrían parecer obsoletos al contemplar los *Mhz* en su descripción. Así, podemos encontrarnos un *HP PA-RISC 8500* a *440 Mhz* con unos resultados muy superiores al de un *P-III Xeon 550 Mhz*.

De hecho a la hora de obtener un *chip* lo más rápido posible, existen dos escuelas:

1. "Haz poco cada vez, pero hazlo muchas veces". Los *chips Alpha 21164* son un ejemplo de este tipo de arquitecturas. Contemporáneos del *Pentium MMX 233*, estos procesadores llegaban a *600 Mhz*.

2. "Aunque no puedas ir tan rápido, haz muchas cosas en cada ciclo de reloj". Los *chips* de *MIPS* y *HP* son un ejemplo de esta filosofía.

A la vista de estas aproximaciones cabría pensar: si diseñamos una *CPU* que ejecute muchas cosas en un ciclo de reloj rápido, tendríamos la solución a todos los problemas. Sin embargo, si queremos hacer demasiadas cosas, tenemos que pasar por un gran número de transistores, cada uno de los cuales retrasa levemente la señal. Si subimos indefinidamente la frecuencia del reloj, llegará un momento en que los impulsos eléctricos se alcanzarán unos a otros y el *chip* dejará de funcionar.

En cuanto a los procesadores compatibles *x86*, hoy en día el mercado está prácticamente copado por *AMD* e *Intel*, con *Rise, IDT* y la desaparecida *Cyrix/National Semiconductor* jugando su papel en ordenadores de muy bajo presupuesto.

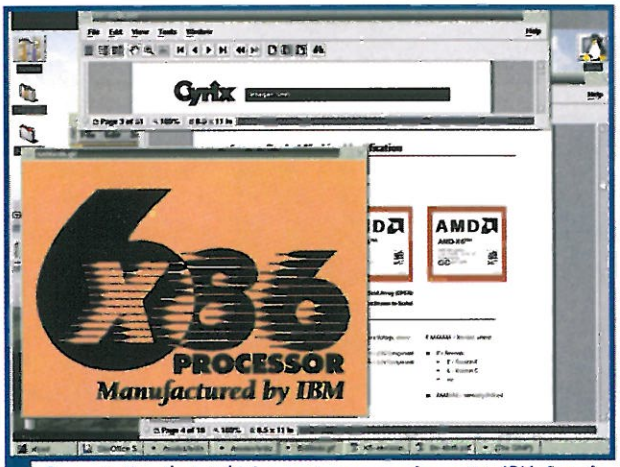


Figura 1.- No sólo *Intel* fabrica microprocesadores *x86*. *IBM, Cyrix/National Semiconductor* y *AMD* también han creado *chips* de calidad.

● Intel ®

Fabrica varios tipos de procesadores para ordenadores de sobremesa, además de una versión de bajo consumo para portátiles. Su oferta permite cubrir las necesidades de todos los segmentos de usuarios con un importante número de versiones.

Basados en la arquitectura del ya veterano *PentiumPro* ® las diferentes *CPU's* que ofrece *Intel* se dividen en tres familias *Celeron* ®, *Pentium* ® y *Xeon* ®

1.- Pentium

Los procesadores de la familia *Pentium*, el *Pentium II* ® y *Pentium III* ®, poseen una arquitectura similar a la del *PentiumPro* ®, con mejoras sobre todo en rendimiento.

Con una caché de segundo nivel de 512 *Kbytes*, funcionando a la mitad de la velocidad del procesador, y *buses* de hasta 100 *Mhz* con la memoria principal. Internamente, estas *CPU's* son un microprocesador *RISC* al que se le añaden un conversor *x86 -> RISC*. Además, estos micros incorporan mejoras como predictores de saltos, numerosas unidades de ejecución, *MMX, Data Forwarding*, Renombrado de Registros y Ejecución Fuera de Secuencia.

La única diferencia entre ambos es la aparición en el *Pentium III* ® de un conjunto de instrucciones destinado a acelerar las operaciones de punto flotante en precisión simple, llamado genéricamente *SSE*. Esta extensión, presente únicamente en el *Pentium III* y el *Pentium III Xeon*, surgió como defensa de *Intel* ante la aparición, varios meses antes, de *3DNow!* de *AMD*.

Las características principales de estos procesadores son: Caché de primer nivel: 16 *Kbytes* Datos / 16 *Kbytes* Instrucciones, extensión *MMX*, caché de 2º nivel: 512 *Kbytes* Unificada (a la mitad de la velocidad del procesador), *bus* principal de 64 *bits* a 100 *Mhz*, soporte para multiprocesador, en el *Pentium III* ®, extensión *SSE*.

2.- Celeron

Opción de bajo coste de *Intel* para competir con la familia *K6* de *AMD*.

Los primeros *Celeron* cobraron una pésima fama, al ser una versión del *PentiumII* ® a la que se le había quitado la caché de segundo nivel. Versiones posteriores, "*Celeron A*", incluyen una caché de segundo nivel de 128 *Kbytes* en la propia *CPU* y funcionando a su velocidad.

Gracias a esta mejora, los procesadores *Celeron* ofrecen una velocidad que oscila entre el 90% y el 95% de la de un *Pentium II*® en la mayoría de aplicaciones ofimáticas y juegos. En programas que utilicen una gran cantidad de datos, como pueden ser las herramientas de manipulación gráfica, las Bases de Datos, etc.., la velocidad puede llegar al 70% de la de un *Pentium II* ® equivalente.

Entre las diferentes características, destacan: Caché de primer nivel: 16 *Kbytes* Datos /16 *Kbytes* Instruc-

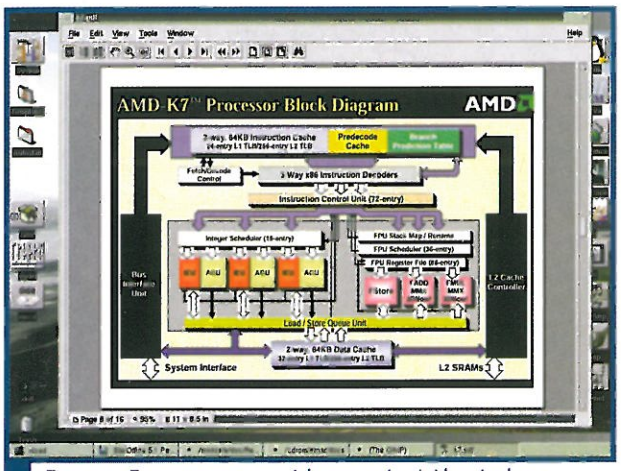


Figura 2.- Estructura interna del procesador Athlon. Incluye: Traductor x86 -" RISC, Unidad de coma flotante (FPU), Unidades MMX/3DNow!, Unidad Load/Store.

ciones, extensión *MMX,* Caché de segundo nivel: 128 *Kbytes* Unificada (a la velocidad del procesador), *Bus* principal de 64 *bits* a 66 *Mhz,* No permite su instalación en sistemas multiprocesador.

3.- Xeon

Gama dirigida a las estaciones profesionales y los servidores corporativos de tamaño medio/bajo. Eran, hasta la aparición del *K7/Athlon*, la opción de mayor rendimiento de toda la familia *x86*.

La única diferencia entre estos procesadores y sus equivalentes *Pentium ® II y III* es la presencia de una caché de 2º nivel, cuyo tamaño varía entre los 512 *Kbytes* y los 2 *Mbytes,* que funciona a la misma velocidad que la *CPU*. Lo que supone ventajas de entre el 5%-10% de mejora en aplicaciones ofimáticas convencionales, hasta un 30% en Bases de Datos u otros programas que se beneficien de una caché de estas dimensiones.

Sus datos relevantes son: Caché de primer nivel: 16 *Kbytes* Datos/16 *Kbytes* instrucciones, extensión *MMX,* caché de 2º nivel: 512/1024/2048 *Kb* unificada (a la velocidad del procesador), *bus* principal de 64 *bits* a 100 *Mhz*, permite su instalación en sistemas multiprocesador, extensión *SSE* en *Pentium III ® Xeon*.

● AMD

Advanced Micro Devices (AMD) ofrece también una amplia gama de procesadores, capaz de satisfacer todos los gustos. Actualmente incluye: *AMD K6-2 ®*, *AMD K6-3 ®* y *AMD K7 Athlon ®*

1.- K6-2

Evolución del procesador *K6*, *CPU* superescalar, con un corazón *RISC* al que se ha acoplado un traductor *x86 -> Risc86*. En ella nos encontramos todas las características de los micros modernos, incluyendo: Ejecución fuera de orden, *predictor* de bifurcaciones, precarga de datos, *write combining*, etc...

El *K6-2 ®* fue el primer procesador del mercado en incorporar una extensión vectorial para aumentar la velocidad de las operaciones en punto flotante de simple precisión, muy importante tanto en las aplicaciones profesionales *3D* como en los juegos de inmersión en primera persona, con la extensión *3DNow!*

Sus características fundamentales son: Caché de primer nivel: 32 *Kb* Datos/32 *Kbytes* Instrucciones, extensión *MMX,* extensión *3DNow!,* Caché de segundo nivel: 512/1024/2048 *Kbytes* unificada (a 100 *Mhz*), *bus* principal de 64 *bits* a 100 *Mhz*, no permite su instalación en sistemas multiprocesador.

2.- K6-3

Estructura interna similar a la del *K6-2,* incorpora una caché de segundo nivel de 256 *Kbytes* con la misma velocidad que el procesador.

K6-3 ofrece una potencia de cálculo para aplicaciones ofimáticas, basadas en números enteros, que supera entre un 10% y un 15% al *Pentium II* ó *III* equivalente. En las operaciones en punto flotante, sin embargo, aún permanece por detrás de las *CPU's* de *Intel*.

Entre sus características destacamos: caché de primer nivel: 32 *Kbytes* Datos/32 *Kb* instrucciones, extensiones *MMX* y *3DNow!,* caché de segundo nivel: 256 *Kbytes* unificada (a la velocidad del procesador), caché de tercer nivel: 512/1024/2048 *Kbytes* unificada (a 100 *Mhz*), *bus* principal de 64 *bits* a 100 *Mhz*, no permite su instalación en sistemas multiprocesador.

3.- Athlon

También conocido como *K7*. Está destinado a servidores corporativos. Al núcleo *RISC* le han añadido un decodificador *x86 -> Risc86*.

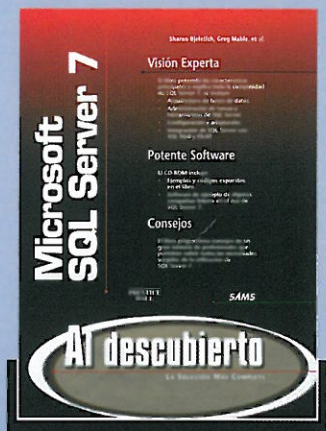
Diseñado para alcanzar rápidamente grandes velocidades de reloj, el *Athlon* ha debutado en versiones de 550, 600 y 650*Mhz*. Este procesador es un ejemplo claro de por qué los *Mhz* no son un indicativo fiable del rendimiento del mismo.

En coma flotante, sin utilizar *3DNow!*, el *K7* ofrece rendimientos que superan en un 35% al *Pentium III ®* y en un 30% al *P-III Xeon* equivalente. En proceso entero, la diferencia es inferior, aunque el *Athlon* aventaja a sus perseguidores en un 15% aproximadamente (siempre a la misma velocidad de reloj).

Sus características incluyen:

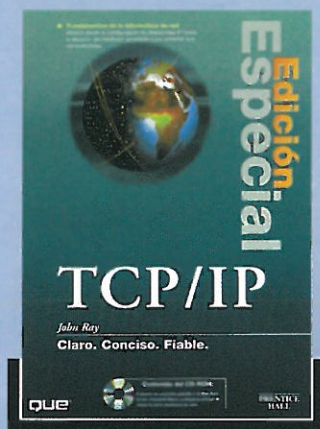
- Caché de primer nivel: 64 *Kbytes* Datos / 64 *Kbytes* Instrucciones.
- Extensión *MMX*.
- Extensión *3DNow!* Mejorada.
- Caché de segundo nivel: 1024 - 8192 *Kbytes* Unificada.
- *Bus* principal *EV6* a 200 *Mhz*.
- Soporte para multiprocesador.

Libros para profesionales escritos por profesionales



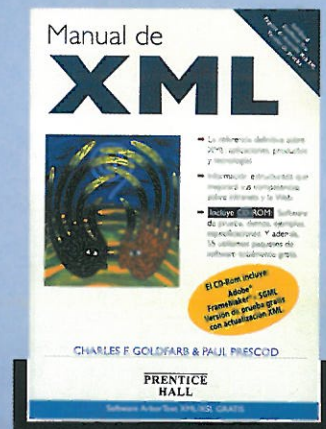
Incluye CD-Rom

Microsoft SQL Server 7 Al Descubierto
Sharon Bjeletich et al.



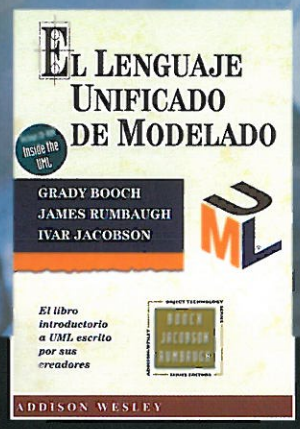
Incluye CD-Rom

Edición Especial TCP/IP
John Ray



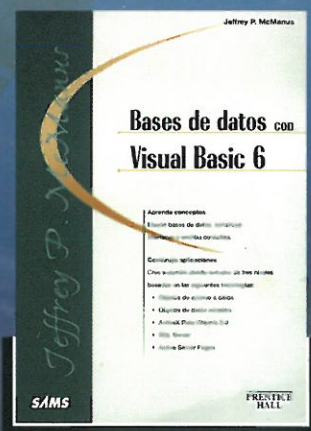
Incluye CD-Rom

Manual de XML
Charles F. Goldfarb & Paul Prescod



Incluye CD-Rom

UML. El Lenguaje Unificado de Modelado
Grady Booch et al.



Incluye CD-Rom

Bases de Datos con Visual Basic 6
Jeffrey P. McManus

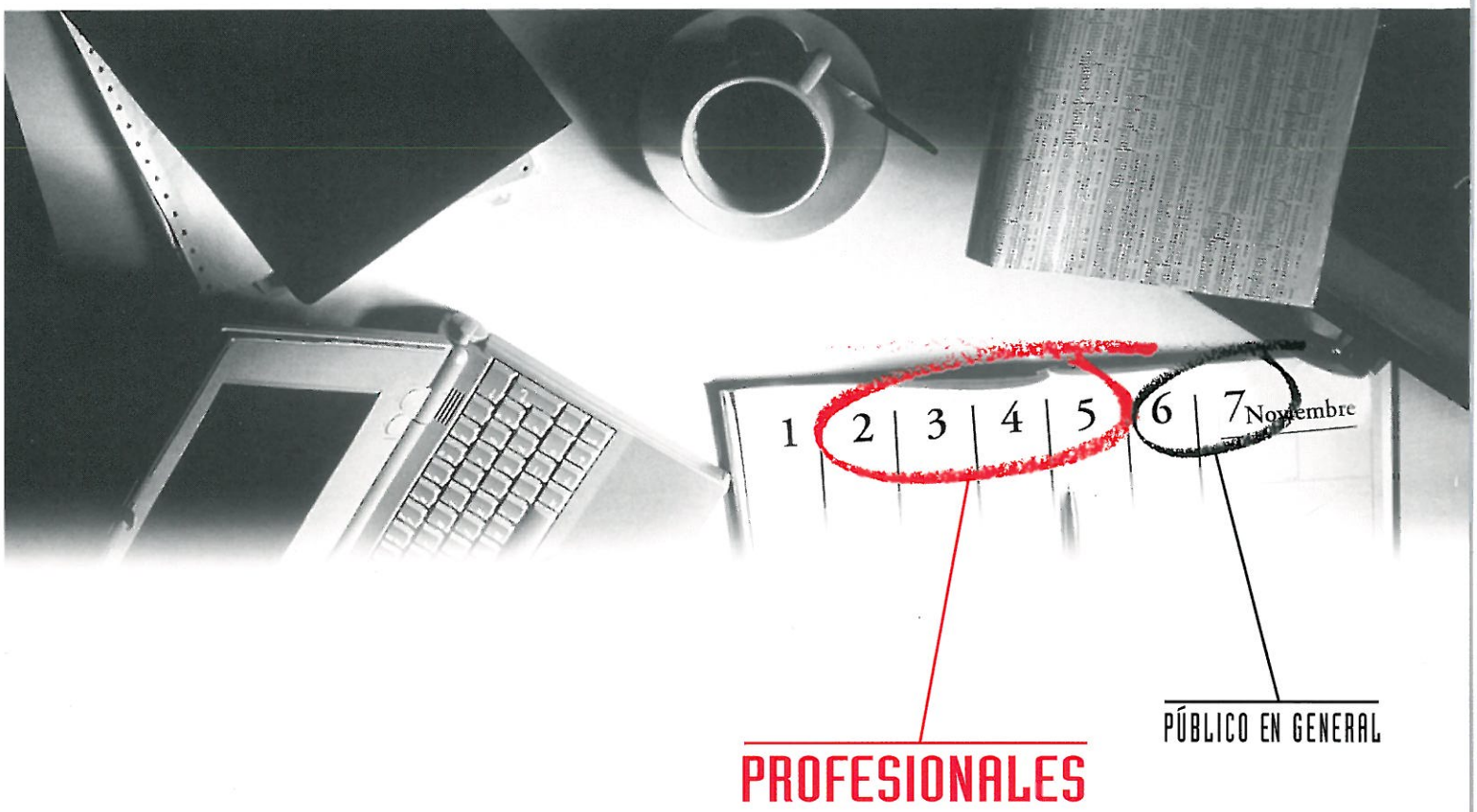
¡Ya a la venta en tu librería habitual!

¡Visítanos en el SIMO!
Pabellón 2
Stand 2038
2-7 de noviembre de 1999

Prentice Hall

Líder mundial en libros y recursos de informática

LOS PROFESIONALES EXIGENTES SÓLO NECESITAN QUE LES RECORDEMOS UNA FECHA



SIMO TCI

Feria Internacional de Informática, Multimedia y Comunicaciones

Madrid, 2-7 Noviembre de 1999

Parque Ferial Juan Carlos I • 28042 Madrid
Apdo. de Correos 67.067 • 28080 Madrid
Tel.: 91 722 51 80 • Fax: 91 722 50 36
e-mail: simo@ifema.es • www.simo.ifema.es

IBERIA
TRANSPORTISTA OFICIAL

IFEMA
Feria de Madrid